TEKNILLINEN KORKEAKOULU

Sähkö- ja tietoliikennetekniikan osasto

Mikko Hämäläinen

HAJAUTETUN HISSINOHJAUSJÄRJESTELMÄN SIMULOINTIYMPÄRISTÖ

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 5.5.2006

Työn valvoja Professori Seppo Ovaska

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Mikko Hämäläinen
Työn nimi:	Hajautetun hissinohjausjärjestelmän simulointiympäristö
Päivämäärä:	5.5.2006 Sivumäärä: 148
Osasto:	Sähkö- ja tietoliikennetekniikan osasto
Professuuri:	S-81 Teollisuuselektroniikka
Työn valvoja:	Professori Seppo Ovaska
Työn ohjaaja:	Professori Seppo Ovaska

Diplomityö valmistelee sulautettujen järjestelmien työkurssin, joka tarjoaa suorittajilleen käytännön kokemusta laitteistoläheisestä ohjelmoinnista reaaliaikaiseen ympäristöön. Kurssin opetuslaitteistona toimii hissinohjausjärjestelmän simulointiympäristö, joka koostuu hissiryhmän signalointielektroniikasta ja kutsunantopaneeleista. Kurssilla kehitettävät sovellusohjelmat siirretään LonWorks-tekniikkaa toteuttavaan Neuron-piiriin, joka mahdollistaa kommunikaation Lon-kenttäväylää pitkin.

Diplomityössä määritellään työkurssin laitekokonaisuus yhdessä laitetoimittajan kanssa, valitaan laitteistosta kontrolleripiirit tarkemman syventymisen kohteiksi, suunnitellaan näille sovellusohjelmat ja muokataan nämä työkurssin harjoitustöiksi.

Opetuslaitteiston näyttävyys ja havainnollisuus saavutetaan näyttöjen ja merkkivalojen avulla. Ne tarjoavat ohjelmoijalle palautetta kehitettävän sovelluksen toiminnasta ja simuloitujen hissien liiketiloista. Laaditut harjoitustyöt sisältävät erilaisia vaikeusasteita, jolloin tehtävänannon pakollinen osa määrittelee sovelluksen perustoiminnallisuuden, jota voidaan vapaaehtoisesti täydentää lisäominaisuuksilla.

Harjoitustöiden laaja-alaisuuteen pyritään keskittymällä järjestelmän yleiskäyttöisiin osakokonaisuuksiin kuten näyttöjen ja painonappien ohjaukseen sekä verkkoliikenteen rajapintoihin. Kenttäväylän viestinnässä tutustutaan verkkomuuttujien lisäksi sovellusohjelmassa eksplisiittisesti muodostettaviin ja lähetettäviin viesteihin, joita käytetään LonWorksin lisäksi muissakin kenttäväylätekniikoissa.

Opetuslaitteiston pääosa edustaa räätälöityä valmistajakohtaista erityisratkaisua, joka ei noudata verkkoliikenteen rajapinnan standardeja. Työkurssilla tutustutaan myös standardin verkkoliikenteen rajapinnan muodostamiseen tätä varten kehitetyssä laitteiston erillisessä osassa, jolloin rajapinnan toteuttava ohjelmakoodi generoidaan automaattisesti ohjelmointityökalulla.

Avainsanat: Avoin järjestelmä, hajautettu järjestelmä, hissinohjausjärjestelmä, kenttäväylä, LonWorks, Neuron C, simulointiympäristö, sulautettu järjestelmä

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Mikko Hämäläinen
Title of the Thesis:	Simulation Environment for a Distributed Elevator Control System
Date:	5.5.2006 Number of Pages: 148
Department:	Electrical and Communications Engineering
Professorship:	S-81 Industrial Electronics
Supervisor:	Professor Seppo Ovaska
Instructor:	Professor Seppo Ovaska

This thesis prepares a practical study course on embedded systems, which gives hands-on practice for the hardware-near software development in a real-time environment.

The application programs of the study course will be developed for selected microcontrollers in a simulation environment. It includes operating panels and signaling electronics of the distributed control system of an elevator. The technology platform of the application programs is a Neuron chip and the LonWorks field bus.

The simulation environment is defined in co-ordination with the hardware supplier, target devices for application programming are selected and example codes for the target devices are created in the thesis. The example codes are further developed to create the assignments of the study course.

The simulation environment provides the user a feedback of the state of elevators and the application program under development through displays and indicator lights of the press buttons. The assignments include multiple levels of challenges with easier compulsory parts and more difficult optional parts. A compulsory part defines the basic functionality of the application program of a target device.

A wide perspective is maintained in the study course by examining the general multi-purpose parts of a whole system, like control of buttons and displays and utilization of network interfaces. A viewpoint to the communication methods in the field bus is not limited to a special solution of network variable, but also more general approach of explicit application messages is introduced. This method is widely used in other field bus technologies than LonWorks.

The main part of the simulation environment presents a manufacturer-specific solution, which does not conform to any standards in its implementation of the network interface. The simulation equipment includes also a separate part to generate a standard network interface with a programming tool.

Keywords: Elevator control system, distributed control system, embedded system, field bus, LonWorks, Neuron C, open control system, simulation environment

ALKUSANAT

Tämä diplomityö on tehty Teknillisen korkeakoulun Tehoelektroniikan laboratoriossa. Työ valmistelee osaltaan laboratorion tarjoamaa uutta sulautettujen järjestelmien opetuskokonaisuutta.

Työn valvojana ja ohjaajana toimi professori Seppo Ovaska, jota haluan kiittää saamastani erinomaisesta ohjauksesta.

Lausun omasta ja Teknillisen korkeakoulun puolesta kiitokset KONE Oyj:lle hajautettua hissinohjausjärjestelmää simuloivan laitteiston lahjoituksesta, mikä on mahdollistanut tämän diplomityön ja sulautettujen järjestelmien työkurssin järjestämisen. Kiitän myös projektiin osallistunutta KONE Oyj:n henkilökuntaa, erityisesti Kari Suihkosta ja Toni Hirvosta.

Kiitän lehtori Veijo Piikkilää LonWorks-tekniikan käyttöä ja koulutusta koskevista tiedoista ja materiaalista. Kiitokset Jukka Seppälälle harjoitustöiden tehtävänantojen testauksesta ja parannusehdotuksista sekä Jouni Envallille kokeneemman tieteenharjoittajan tuesta ja kommenteista.

Kiitän myös muita ystäviäni kannustuksesta sekä vanhempiani opiskeluaikanani saamastani taloudellisesta ja kaikesta muusta tuesta.

Espoossa 5.5.2006

Mikko Hämäläinen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ANSI	The American National Standards Institute, Yhdysvaltain kansallinen standardointijärjestö
API	Application Programming Interface, sovellusliitäntä- eli ohjelmointirajapinta, jonka avulla sovellus hyödyntää esim. toisen sovelluksen palveluja ohjelmallisesti
C	laitteistoläheinen ohjelmointikieli
CSMA	Carrier Sense Multiple Access, tiedonsiirtoväylän kilpavaraus
CSMA/CD	CSMA / Collission Detection, törmäyksen tunnistus
CRC	Cyclic Redundancy Check, tiedonsiirron bittitason virheentarkistusmenetelmä
DIP	Dual In-line Package, mikropiirin suorakulmainen kotelo, jonka liitinnastat ovat kahdessa rivissä kotelon vastakkaisissa reunoissa
DIP-kytkin	kaksirivikytkin, rivi kaksiasentoisia kytkimiä sijoitettuna DIP-koteloon, sijaitsee piirilevyllä, käytetään laitteiston asetusten tekoon
EEPROM	Electrically Erasable and Programmable Read Only Memory, sähköisesti tyhjennettävä ohjelmoitava lukumuisti, jonka yksittäisiä sanoja voidaan sähköisesti tyhjentää ja ohjelmoida useaan kertaan
flash-muisti	EEPROM-muistin kaltainen sähköisesti tyhjennettävä ohjelmoitava lukumuisti, jossa muistiin kirjoitetaan tuhansien tavujen kokoisissa lohkoissa
ISO	International Organization for Standardization, kansainvälinen standardointiorganisaatio
LED	Light Emitting Diode, valoa lähettävä diodi
Lon	Local Operating Network, eräs kenttäväyläjärjestelmä, erityisesti käytössä kiinteistöhallinnassa
LonTalk	LonWorks -teknologian kommunikointiin tarvittava protokolla
LonWorks	avoin ja hajautettu automaatiojärjestelmä
Neuron C	LonWorks-tekniikan tarpeita varten kehitetty ohjelmointikieli
Neuron-piiri	kolmen prosessorin muodostama mikropiiri, joka sijoitetaan jokaiseen Lon-verkon solmuun, toteuttaa LonTalk-protokollan
OSI	Open Systems Interconnect, ISO:n viitemalli tietoliikenteen yhdenmukaistamiseksi, tietoliikenteen avoin viitemalli (ISO 7498)
RAM	Random Access Memory, suorasaantimuisti, jota käytetään esim. tietokoneen keskusmuistina
ROM	Read Only Memory, lukumuisti, jota ohjelma voi lukea mutta johon ei voi tallentaa tietoa
Solmu	LonWorks-verkkoon liitetty laite, joka viestii muiden solmujen kanssa mutta kykenee myös itsenäiseen toimintaan, engl. ”node”

1 JOHDANTO

Teknillisen korkeakoulun Tehoelektroniikan laboratorion uusi sulautettujen järjestelmien opetuskokonaisuus tarjoaa laaja-alaisen yleiskuvan valmiiden mikropiirien valitsemiseksi ja soveltamiseksi. Tämä diplomityö valmistelee opetuskokonaisuuden sisältävää sulautettujen järjestelmien työkurssia. Työkurssi tarjoaa suorittajilleen käytännön kokemusta laitteistoläheisestä ohjelmoinnista reaaliaikaiseen ympäristöön.

Sulautettujen järjestelmien työkurssin opetuslaitteistona toimii hissinohjaus-järjestelmän simulointiympäristö, joka koostuu hissiryhmän signalointielektroniikasta ja kutsunantopaneeleista. Laitekokonaisuus määriteltiin diplomityötä tehtäessä ja se saatiin lahjoituksena KONE Oyj:ltä. Diplomityöhön on sisältynyt sovellusohjelmiston suunnittelu simulointiympäristön laitteistolle sekä näiden sovellusohjelmien muokkaaminen työkurssin harjoitustöiksi.

Diplomityön tavoitteena on ollut käyttökelpoisen opetuslaitteiston kehittäminen yhteistyössä laitetoimittajan kanssa. Laitteistossa on pyritty näyttävyyteen ja havainnollisuuteen antamalla käyttäjälle palautetta näyttöjen ja merkkivalojen avulla. Harjoitustöiden osalta pyrkimyksenä on ollut tuottaa erilaisia vaikeusasteita sisältäviä tehtäviä, joiden suorittajalle asetetaan vähimmäisvaatimukset ja tarjotaan myös lisähaasteita.

Hajautetussa hissinohjausjärjestelmässä fyysisesti eri paikoissa sijaitsevat laitteet kommunikoivat keskenään kenttäväylän välityksellä järjestelmän tehtäväkokonaisuutta toteuttaen. Työkurssin sovellusohjelmat kirjoitetaan LonWorks-tekniikkaa toteuttavaan Neuron-piiriin, joka mahdollistaa kommunikaation Lon-kenttäväylää pitkin.

Hissinohjausjärjestelmä on sulautettujen järjestelmien erikoisratkaisu, LonWorks on vain eräs kenttäväylätekniikka ja simulointiympäristön laitetoimittajan tapa hyödyntää LonWorks-tekniikkaa on tavanomaisesta poikkeava. Opetusmateriaalia kehitettäessä on tavoitteena ollut laaja-alaisuuden säilyttäminen, mihin on pyritty keskittymällä tarkastelussa yleiskäyttöisiin komponentteihin ja esittelemällä vaihtoehtoisia verkonhallintaohjelmia. On myös pyritty esittelemään vaihtoehtoisia viestinvälitystekniikoita ja verkkoliikenteen rajapintoja.

Työn liitteinä olevien työkurssin harjoitustöiden tehtävänantojen lisäksi diplomityön teoriaosuus toimii työkurssin opetusmateriaalina. Luvussa kaksi tutustutaan Neuron-piiriin ja LonWorks-teknologian tavoitteena olevaan yhteentoimivuuteen. Luvussa kolme tarkastellaan LonTalk-protokollaa, jota noudattaen Neuron-piirin sisältävät laitteet kommunikoivat kenttäväylää pitkin. Teknologiaa ja protokollaa ei käsitellä kattavasti, vaan sovellusohjelmoinnin näkökulmasta.

Neljäs luku toimii työkurssin ohjelmointioppaana, joka opastaa Neuron-piiriä varten kehitetyn Neuron C -ohjelmointikielen käyttöön. Viides ja kuudes luku tarkastelevat Neuron-piirin tietorakenteita. Kuudes luku keskittyy verkkoliikenteen vaatimiin asetuksiin. Niillä on erityinen painoarvo tässä diplomityössä, koska simulointi-ympäristössä verkkoasetukset asetetaan Neuron-piirin sovellusohjelmasta käsin. Teoreettisen tarkastelun painotuksia on ohjannut pyrkimys selvittää kaikki harjoitusten ohjelmakoodissa tarvittavat asetusparametrit. Näin työkurssin suorittaja voi halutessaan vaivattomasti tutustua kaikkiin ohjelmakoodin yksityiskohtiin.

Luvussa seitsemän tarkastellaan tuotettuja harjoitustöitä ja niiden kehittämiseen vaadittua prosessia. Valmistellut harjoitustyöt ovat liitteinä 1. - 5. ja simulointi-ympäristön käyttöohje liitteessä 6.

2 LONWORKS-TEKNIIKKA

Perinteinen keskitetty automaatiojärjestelmä pohjautuu keskustietokoneeseen. Siinä keskusyksikkö kyselee tiloja kenttälaitteilta, jotka kytkeytyvät omilla kaapeleillaan automaatiojärjestelmään. Avoimissa ja hajautetuissa automaatiojärjestelmissä kenttälaitteet kommunikoivat keskenään kenttäväylän avulla. Se mahdollistaa tiedon hajautetun käsittelyn ilman keskitettyä valvontaa. Hajautetussa järjestelmässä säästetään kaapelointi- ja asennuskustannuksissa, kun useamman laitteen välinen tietoliikenne voidaan suorittaa samaa johdotusta pitkin. Säästöt ovat merkittäviä niin autoteollisuudessa kuin rakennusten hissikuiluissakin (Dietrich ym. 2001).

LonWorks-teknologia toteuttaa avoimen ja hajautetun järjestelmän. Echelon Corporation omistaa oikeudet LonWorks-tekniikan pohjana toimivaan Neuron-piiriin ja sovellusohjelman luomiseen käytettyihin työkaluihin. Nimi LonWorks kuvaa koko järjestelmän toteuttamiseen tarvittavaa teknologiaa, johon sisältyvät myös LonTalk-protokolla, lähetin-vastaanottimet sekä verkonhallintaohjelmat (Graham 2006).

LonWorks-tekniikassa Lon-kenttäväylään on kytketty itsenäiseen toimintaan kykeneviä laitteita eli verkon solmuja, jotka kommunikoivat keskenään tiedon siirtämiseksi ja resurssien jakamiseksi. Näin erilliset laitteet voivat yhdessä toteuttaa järjestelmän tehtäväkokonaisuutta, ja hajautetun rakenteen ansiosta järjestelmää on helppo mukauttaa muuttuviin tarpeisiin eikä järjestelmän toiminta pysähdy yhden laitteen hajoamiseen. Myös väylän kapasiteetti tulee käytettyä keskitetysti ohjattua järjestelmää tehokkaammin, kun hajautetun järjestelmän älykkäät solmut lähettävät tietoa vain tarpeen mukaan, eikä raskasta kiertokyselymenetelmää tarvita (Piikkilä 2004).

Jokainen Lon-verkon solmu sisältää Neuron-piirin, jonka sovellusohjelma kommunikoi muiden solmujen sovellusohjelmien kanssa. Kommunikaatio tapahtuu LonTalk-protokollan välityksellä, jonka Neuron-piirin varusohjelmisto yhdessä kolmen 8-bittisen prosessorin kanssa toteuttavat.

Neuron-piirin lisäksi Lon-verkon solmu sisältää lähetin-vastaanottimen, joka mahdollistaa fyysisen yhteyden kenttäväylään. Lähetin-vastaanottimen tyyppi määräytyy käytettävän siirtotien perusteella, ja siirtotienä voi toimia kierretty parikaapeli, sähköverkon johdotus, radioaallot, infrapuna, koaksiaalikaapeli sekä valokuitu.

Neuron-piiriä ohjaava sovellusohjelma kirjoitetaan aina Echelonin kehittämällä ohjelmointityökalulla. Yksittäisen solmun kehitykseen on tarjolla NodeBuilder-ohjelmisto. Se sisältää editorin ohjelmakoodin kirjoittamiseen ja sen mukana toimitetaan myös LonMaker-verkonhallintaohjelma. Laajempien järjestelmien kehitykseen on tarjolla LonBuilder-ohjelmointiympäristö, joka sisältää mm. protokolla-analysaattorin sekä tuen reitittimille.

2.1 Yhteentoimivuus

LonWorks-tekniikassa eri laitteiden verkkoliikenteen rajapinnan objektit voidaan kytkeä toisiinsa, kunhan niillä on yhtenevät tietotyypit. Tietotyyppien käytön yhtenäistämiseksi on perustettu avoin organisaatio hallinnoimaan niiden standardointia. Yhteentoimivuuden perusedellytyksenä on laitteiden kommunikaatio-rajapintojen selkeä ja yhdenmukainen määrittely. LonMark-organisaatio on määritellyt standardeja toiminnallisia profiileja eri sovelluksille ja eri teollisuuden alojen tarpeisiin (LonMark 2006). Toiminnallinen profiili sisältää kokoelman standardeja verkkomuuttujia ja asetusparametreja tietyn toiminnan suorittamiseen. Standardit verkkomuuttujat sisältävät määrittelyn muuttujan koodaukselle,

skaalaukselle ja yksiköille. Niitä on määritelty lähes kaikille fysikaalisille suureille sekä myös abstrakteimmille käsitteille eri teollisuuden alojen sovelluksiin. Esim. LVI-järjestelmien säätöön on verkkomuuttujat järjestelmän toimintatilan asettamiseen ja viestittämiseen, tehdyn asetuksen syrjäyttämiseen (override) sekä hätätilan toimintojen asettamiseen.

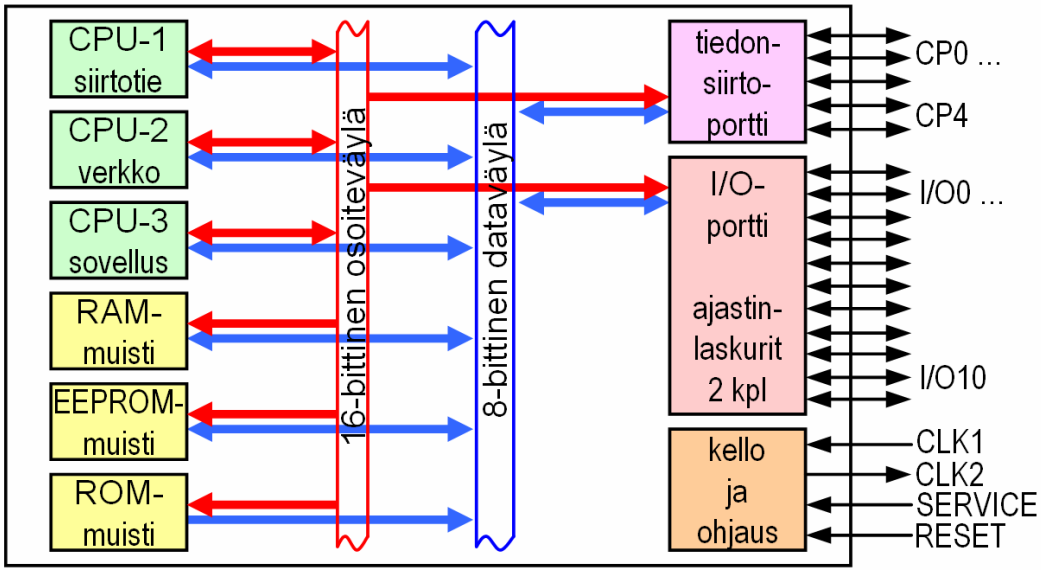
Yhteentoimivuus helpottaa laitteiden asennusta erilaisiin verkkoihin, kun laitteen tiedonsiirron rajapinnan rakenne on riippumaton laitteen sovelluskentästä. LonWorks-tekniikassa on onnistuttu eriyttämään solmun sovellusohjelman toiminta kohdeosoitteiden ja loogisten kytkentöjen määrittelyistä. Tämä on tärkeä askel eri valmistajien tuotteiden väliseen yhteentoimivuuteen, kun laitteiden väliset kytkennät voidaan tehdä käyttökohteessa laitteen sisältämästä sovellusohjelmasta riippumatta. Laitteiden välisen rajapinnan komponentit täytyy toki sopia yhteisesti, ja juuri tätä varten LonMark ylläpitää standardeja verkkomuuttujia ja toiminnallisia profiileja. Kun verkko-osoitteiden ja -palvelujen yksityiskohdat voidaan määritellä käyttökohteessa verkonhallintaohjelmalla, voidaan verkkoa myös jatkossa joustavasti ylläpitää ja muunnella tarpeiden mukaisesti.

2.2 Neuron-piiri

LonWorks-tekniikan pohjan muodostaa Neuron-piiri, jonka sisältämä varusohjelmisto huolehtii tiedonsiirrosta LonTalk-protokollaa käyttäen. Neuron-piiri sisältää kolme kahdeksanbittistä prosessoria, jotka toimivat rinnakkain eri tehtäviä suorittaen. Siirtotie- ja verkkoprosessorit mahdollistavat solmujen välisen tietoliikenteen Lon-verkkossa, ja sovellusprosessori suorittaa sovellusohjelman ohjaamana solmun paikallista tehtävää, esim. mittausta tai toimilaitteen säätöä. Neuron-piirin rakenne on esitetty kuvassa 1.

Neuron-piiri kytkeytyy viidellä pinnillä lähetin-vastaanottimeen, joka mahdollistaa fyysisen yhteyden kenttäväylään. Neuron-piirin 11 pinnin I/O-rajapinta mahdollistaa esim. moottorin, näytön, A/D-muuntimen, anturin tai releen ohjaamisen.

Neuron-piiri sisältää kaksi 16-bittistä ajastin/laskuria, joiden sisään- ja ulostulot ovat kytkettävissä I/O-portteihin. Ajastimen kellosignaali voidaan lukea I/O-pinnin kautta tai kellosignaalina voidaan käyttää skaalattua järjestelmän kellosignaalia. Huolto-kytkin (Service pin) mahdollistaa asennuksen yhteydessä tapahtuvan Neuron-piirin tunnistamisen Lon-verkossa (Motorola 1997b).



Kuva 1. Neuron-piirin rakenne.

Pelkkää Neuron-piiriä käytettäessä sovellusohjelman suorittamiseen on tarjolla melko vaatimattomat laitteistoresurssit. Esimerkiksi Toshiban 3120FE5M-piiri toimii korkeintaan 20 MHz kellotaajuudella ja sisältää 4 Kb SRAM- ja 3 Kb EEPROM-muistia. 16 Kb:n ROM-muisti sisältää varusohjelmiston (Toshiba 2001).

Sovellusohjelman ohjelmakoodia tai dataa varten saadaan lisää tilaa käyttämällä Neuron 3150 -piiriä, johon liitetään ulkoinen ROM- tai flash-muistipiiri. Neuron-piirin 16-bittinen osoiteavaruus mahdollistaa ulkoisten muistialueiden osoittamisen 42 kilotavuun asti. Sama muistiavaruus kattaa ROM-, EEPROM- ja RAM-muistit. Sovellusohjelmassa yhdellä tavulla voidaan osoittaa kunkin muistityypin lähimuistialuetta eli ensimmäistä 256 tavua. Määritettäessä verkkomuuttujia lähialueen ulkopuolelle käytetään far-määrettä. Neuron-piiriin sovellusohjelmaa kirjoitettaessa muistin rajallisuus aiheuttaa sen käytön optimoinnin tarpeen.

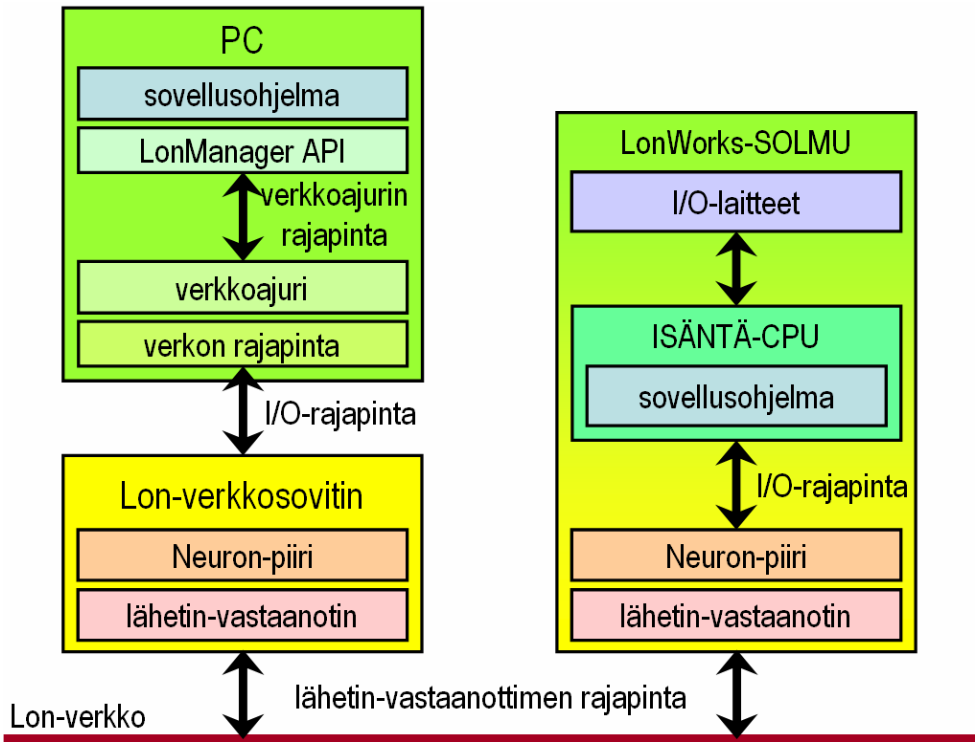
Ulkoista muistia käyttävässä Neuron 3150 -piirissä sovellusohjelma sijaitsee yleensä ulkoisessa ROM- tai flash-muistissa, jos sen kapasiteetti on riittävä. Ulkoiselle muistipiirille sijoitetaan kuitenkin ensisijaisesti varusohjelmisto, koska se ei voi sijaita integroidussa EEPROM-muistissa sen hitauden takia. Integroituun EEPROM-muistiin kirjoittaminen vaatii 20 ms tavua kohti, eikä muistia voida tänä aikana lukea. Tämä viive on liian pitkä Neuron-piirissä sovellusprosessorin rinnalla toimiville siirtotie- ja verkkoprosessoreille, jotka tarvitsevat varusohjelmiston palveluja myös EEPROM-muistin kirjoituksen aikana (Cypress 2000). Atmelin ulkoiset flash-muistipiirit sisältävät kirjoituspuskureita, joiden ansiosta kirjoittaminen ei varaa osoite- ja dataväylää tarpeettomasti (Atmel 2004). Näin myös flash-muistissa sijaitseva varusohjelmisto on koko ajan Neuron-piirin käyttöjärjestelmän tavoitettavissa.

Echelon julkaisi ensimmäiset LonWorks-tuotteet joulukuussa 1990. Echelonin rahoittajina toimivat amerikkalaiset riskisijoittajat yhdessä Motorolan ja Toshiban kanssa. Nämä puolijohdevalmistajat myös valmistivat Neuron-piirejä Echelonin lisenssillä, ja Motorolan vetäydyttyä valmistuksesta tuli Cypress Semiconductor Toshiban rinnalle Neuron-piirien valmistajaksi (Belimo 2005). Toshiba ja Cypress ostivat kymmenen vuoden valmistuslisenssin Echelonilta vuosituhannen vaihteessa (Graham 2006). Echelon myös teettää Smart Transceiver -piirejä, jotka se markkinoi itse. Nämä piirit sisältävät integroidun lähetin-vastaanottimen (Echelon 2004).

2.2.1 LonWorks-sovellukset erillistä isäntäprosessoria käyttäen

Sovellusohjelman vaatiessa enemmän laskutehoa, muistia tai laajemmat I/O-ominaisuudet kuin mitä Neuron-piiri kykenee tarjoamaan, siirretään sovellusohjelman suoritus PC:lle tai muulle erilliselle isäntäprosessorille kuvan 2. mukaisesti (Echelon 1993). Tämä mahdollistaa sovellusohjelman kehityksen halutussa ohjelmointiympäristössä sekä jo olemassa olevan tuotteen kytkemisen Lon-verkkoon vähäisin suunnittelukustannuksin. Lisäksi erillinen isäntäprosessori mahdollistaa LonWorks-sovellukselle jopa 4096 verkkomuuttujaa, kun itsenäisesti toimiva Neuron-piiri kykenee vain 62 verkkomuuttujan käsittelyyn. Verkkomuuttujien avulla tehtävien loogisten kytkentöjen suuri kapasiteetti on hyödyksi valvonta-, tietojenkeruu- sekä ohjainsovelluksissa.

Erillistä isäntäprosessoria käytettäessä Neuron-piiri toimii kommunikaatioprosessorina, jonka välityksellä isäntäprosessori kommunikoi Lon-verkon muiden solmujen kanssa LonTalk-protokollaa käyttäen. LonTalk-protokolla sisältää määrittelyn isäntäprosessorin ja kommunikaatioprosessorina toimivan Neuron-piirin väliselle viestinnälle.



Kuva 2. LonWorks-sovellusta voidaan ajaa myös Neuron-piirin ulkopuolella.

Sovellusohjelma voidaan tehdä PC:lle, jolloin PC kytkeytyy Lon-verkkoon verkkosovittimen välityksellä. Echelon valmistaa eri liityntätapoja käyttäviä verkkosovittimia. Tarjolla on PC:n ISA- ja PCI-väylää, sarjaporttia, PC-korttipaikkaa sekä internet-yhteyttä käyttäviä sovittimia (Echelon 2003c). Verkkosovittimen Neuron-piiri ja lähetin-vastaanotin toteuttavat viisi alinta LonTalk-protokollan kerrosta, jolloin kaksi ylintä jäävät PC:ssä ajettavalle sovellusohjelmalle.

LonManager-sovellusliittymän käyttö helpottaa sovelluksen luomista PC:lle. Laitteistosta riippuvainen verkon rajapinnan protokollan osa voidaan toteuttaa verkkoajurilla, jolloin varsinainen sovellusohjelma on riippumaton PC:n ja verkkosovittimen välisestä fyysisestä rajapinnasta.

Tyypillisesti verkonhallinta- ja valvontaohjelmat toteutetaan PC-sovelluksina. PC:n käyttö mahdollistaa monipuolisten käyttöliittymien ja tallennusmenetelmien käytön sovellusohjelmassa. Lisäksi valmiita muiden valmistajien laitteita ja ohjelmistoja voidaan kytkeä PC:n ja verkkosovittimen välityksellä Lon-verkkoon.

PC:n lisäksi myös muu mikroprosessori tai -kontrolleri voi toimia LonWorks-sovelluksen isäntänä. Sille voidaan rakentaa samaan laitteeseen integroitu räätälöity verkkosovitin, jolloin isäntäprosessori kommunikoi Neuron-piirin kanssa 11-bittistä rinnakkaisväylää tai kaksiporttista RAM-muistia käyttämällä. Neuron-piirin varusohjelma MIP (Microprocessor Interface Program) mahdollistaa tällöin piirin käytön kommunikaatioprosessorina (Echelon 1995c). MIP toteuttaa Neuron-piirin puoleisen osan isännän ja kommunikaatioprosessorin välisestä protokollasta.

Erillistä isäntäprosessoria käyttämällä voidaan laajentaa LonWorks-laitteen I/O-ominaisuuksia. Tietyille prosessoriperheille on tarjolla runsaasti valmiita oheispiirejä, joista voidaan valita tarvetta vastaava lähetin-vastaanotinpiiri, analogia-digitaalimuunnin tai ohjainpiiri. Ohjainpiiriä voidaan tarvita esim. laitteen sisältämän näytön, näppäimistön tai moottorin ohjaukseen.

Motorola valmisti Neuron-piirejä ennen vuosituhannen vaihdetta, mikä antoi motivaation kehittää Neuron-piirien yhteiskäyttöä yhtiön muiden piirien kanssa. MC68332-prosessorille ehdittiin luoda C-kieliset esimerkkiajurit sen liittämiseksi Neuron-piiriin rinnakkaisväylää käyttäen. Esimerkkiajurit mahdollistavat nopean tuotekehityksen ja pienillä muutoksilla niitä voidaan soveltaa koko MC68000-prosessoriperheeseen (Motorola 1997c).

3 LONTALK-TIEDONSIIRTOPROTOKOLLA

Jokainen Lon-verkon solmu sisältää Neuron-piirin, jonka sovellusohjelma kommunikoi muiden solmujen sovellusohjelmien kanssa. Kommunikaatio tapahtuu LonTalk-protokollan välityksellä, jonka Neuron-piirin varusohjelmisto yhdessä kolmen kahdeksanbittisen prosessorin kanssa toteuttavat. Protokolla näkyy sovellusohjelmalle kokoelmana palveluita ja funktioita, jotka sisältyvät Neuron-piirin valmisohjelmistoon. Valmisohjelmiston funktiot hoitavat siis solmujen välisen tietoliikenteen Lon-verkkossa, ja sovellusohjelmassa voidaan keskittyä solmun paikallisen tehtävän hoitamiseen, esim. mittaukseen tai toimilaitteen säätöön.

LonTalk-protokolla noudattaa läheisesti ISO:n OSI-mallia (ISO 1981), ja taulukossa 1. on verrattu LonTalkin tiedonsiirtopalveluiden toteutustasoja OSI:n protokollapinon kerroksiin (Piikkilä 2002). Taulukko eroaa sovellusprosessorin tehtävien laajuuden osalta Toshiban tuotejulkaisusta (Toshiba 2006).

Taulukko 1. LonTalk-protokollan OSI-malliin verrattavat kerrokset.

	OSI-kerros	Tehtävät	Palvelut	CPU
7	sovellus	sovelluksen yhteensopivuus	LonMark-objektit, standardit verkkomuuttujat	sovellus
6	esitystapa	tiedon tulkinta	verkkomuuttujat, sovellusviestit, verkonhallintakäskyt	sovellus
5	yhteysjakso	etätoiminnot	kysely/vastauspalvelu	verkko
4	kuljetus	päästä-päähän luotettavuus	kuitatut, toistetut ja kertaviestit, aidonnus, pakettien järjestys, kaksoiskappaleiden havainnointi	verkko
3	verkko	kohteen osoitus	solmu-, ryhmä- ja yleislähetys, reititys	verkko
2	siirtoyhteys	siirtotien varaus ja kehystys	kehystys, koodaus, CRC, CSMA, prioriteetti, törmäysten havaitseminen	siirtotie
1	fyysinen	sähköinen kytkentä	tiedonsiirtomedian mukainen rajapinta ja modulaatio	siirtotie, lähetin-vastaanotin

3.1 Tiedonsiirtomediat

LonTalk-protokolla on riippumaton siirtoon käytetystä mediasta ja siirto on mahdollista mm. kierrettyä parikaapelia, verkkosähköjohdotusta, radioaaltoja, infrapunaa, koaksiaalikaapelia ja valokuitua pitkin. Tiedonsiirtomedia määrää siirtonopeuden ja väylällä käytetyn modulaatiomenetelmän. Jokainen Lon-verkon solmu liittyy väylälle lähetin-vastaanottimen välityksellä, joka moduloi lähetettävän datan siirtomedialle sopivaksi. Lähetin-vastaanotin voi olla puhtaasti erillinen piiri tai se voi olla osittain integroituna Neuron-piiriin, jolloin tuotteen markkinointinimenä on Smart Transceiver (Echelon 2004).

LonTalk-protokolla tukee useita tiedonsiirtokanavia, jotka voivat koostua eri tiedonsiirtomedioista. Reititin sisältää lähetin-vastaanottimen kumpaankin kanavaan, ja mahdollistaa paketin siirtymisen kanavasta toiseen. Reititin myös suodattaa pois yhdistämiensä kanavien sisäisen liikenteen, jolloin paikalliset viestit eivät kuormita koko verkkoa.

3.2 Liikennemääriin sopeutuva kilpavaraus

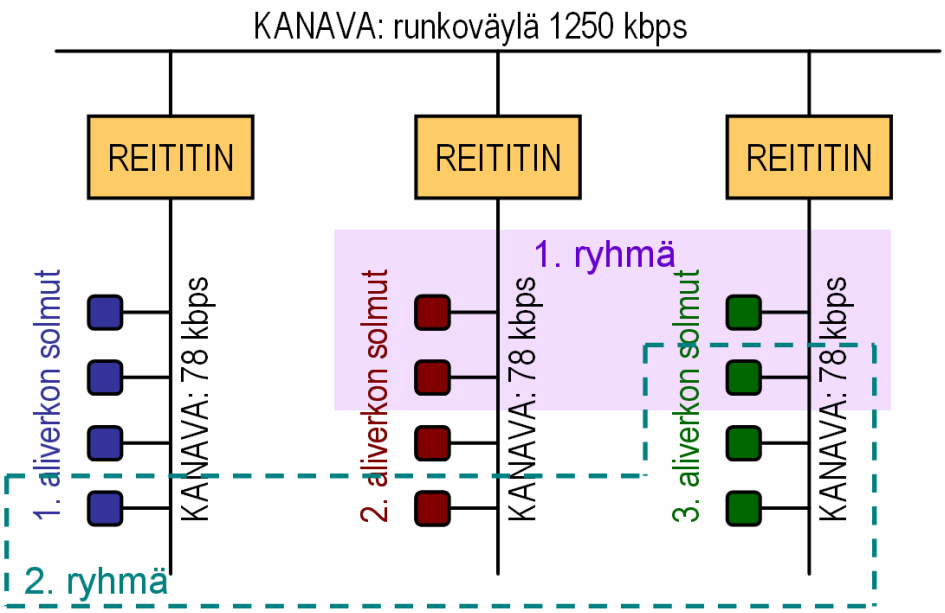
Neuron-piirin siirtotieprosessori kehystää paketit ja koodaa niiden sisältämän datan, suorittaa CRC-virheentarkistuksen ja havaitsee törmäykset. Siirtotien varauksessa se käyttää kehittynyttä kilpavarausperiaatetta, predictive-persistent CSMA:ta. Menetelmässä lähettävä solmu kuuntelee väylää ja odottaa sen vapautumista. Sitten se arpoo lähetyshetken annetusta aikaikkunoiden joukosta. Niiden lukumäärää lisätään verkkoa kuormitettaessa ja näin minimoidaan törmäysten todennäköisyys. Liikennemääriin sopeutuminen aikaikkunoiden määrää säätämällä voidaan tehdä ennakoiden, sillä vähintään puolet verkon liikenteestä on ennakoitavissa kuittaavaa tiedonsiirtopalvelua käytettäessä.

Kaikki solmut ylläpitävät tietoa verkon kuormituksesta. Viestin lähettävä solmu liittää viestiin tiedon sen aiheuttamien vastausviestien oletetusta lukumäärästä. Kaikki viestin kuulevat solmut lisäävät tämän luvun verkonkuormitustietoonsa. Vastaavasti viestin aiheuttamien kuittausten jälkeen kuormitustieto jälleen pienenee (Piikkilä 2002).

LonTalk-protokollassa solmulle voidaan lisäksi asettaa prioriteettiaikaikkuna, joka on voimassa tiettynä hetkenä jokaisen kanavassa lähetetyn viestin jälkeen. Tänä aikana muut solmut eivät saa aloittaa lähetystä, joten prioriteettiaikaikkunaa käyttämällä siirtotien varaus onnistuu nopeammin.

3.3 Lon-verkon hierarkia

Verkkoprosessori vastaa kaikista ylemmistä protokollapinon kerroksista lukuun ottamatta sovellustasoa, jota varten on oma prosessorinsa. Verkkokerroksessa lähetettävä paketti reititetään ja varustettaan kohteen mukaisella osoitetyypillä. Osoite voidaan kohdistaa yhteen solmuun, ryhmään tai koko pää- tai aliverkkoon.



Kuva 3. Pääverkon solmut ovat aliverkkojen ja ryhmien jäseniä.

Ylimpänä LonTalk-protokollan osoitteenmuodostuksen hierarkiassa on pääverkko. Sen tunniste voi olla enintään kuuden tavun pituinen ja yksi solmu voi kuulua korkeintaan kahteen pääverkkoon. Samassa tiedonsiirtokanavassa toimivat erilliset verkkosovellukset ja niiden alaiset laitteet voidaan erottaa toisistaan eri pääverkon tunnuksella.

Seuraava taso osoitteenmuodostuksessa on aliverkko, joka on looginen kokoelma solmuja yhdestä tai useammasta kanavasta. Yhdessä pääverkossa voi olla korkeintaan 255 aliverkkoa. Älykäs reititin toimii aliverkkojen tasolla. Se tunnistaa molemmilta puoliltaan löytyvät aliverkot ja läpäisee paketteja vastaavasti (Toshiba 2006).

Kolmas osoitteenmuodostuksen taso on solmu, joita yksi aliverkko voi sisältää 127 kappaletta. Näin yksi pääverkko voi sisältää jopa 32385 solmua. Kuuluessaan kahteen eri pääverkkoon solmu voi toimia niiden välisenä yhdyskäytävänä.

Solmut voivat myös kuulua ryhmiin, joita voi yhdessä pääverkossa olla 256 kappaletta. Saman ryhmän jäsenet voivat kuulua eri kanaviin ja aliverkkoihin. Ryhmäosoitteet vähentävät bittien lukumäärää osoitetiedoista ja välittävät viestin usealle solmulle yhdellä viestillä. Yksi solmu voi kuulua korkeintaan 15 eri ryhmään.

Jokainen solmu liittyy kanavaan. Se on reitittimillä jaettu verkon osa, johon liitetyt laitteet käyttävät samaa tiedonsiirtonopeutta. Kanava saattaa koostua toistimilla yhdistetyistä erillisistä väyläkaapeleista eli segmenteistä, mutta loogisesti yhtenäisenä verkon osana se välittää kaikki sisältämiensä solmujen lähettämät viestit. Erinopeuksiset ja eri tiedonsiirtomedioita käyttävät kanavat yhdistetään toisiinsa reitittimillä. Kuvaan 3. on merkitty tyypillisiä Lon-verkon siirtonopeuksia, joskaan teknologia ei ole sidoksissa tiettyihin siirtonopeuden arvoihin.

Pääverkon sisältämiä aliverkkoja yhdistää usein nopea runkoväylä, johon aliverkkojen kanavat kytkeytyvät reitittimien välityksellä. Runkoväylän kapasiteettia voidaan säästää sijoittamalla keskenään runsaasti kommunikoivat solmut samaan kanavaan ja määrittelemällä sen sisältämät solmut omaan aliverkkoonsa. Tällöin reititin suodattaa kanavan sisäiset viestit runkoväylään kytkeytyvästä liikenteestä. Runkoväylässä kulkevat toisille aliverkoille tarkoitetut viestit eivät myöskään kuormita reitittimen suojaamaa aliverkkoa.

3.4 Tiedonsiirron palvelumuodot

Tiedonsiirtoon on tarjolla neljä eri palvelumuotoa, joista protokollapinon kuljetuskerros suorittaa kuitatut, toistetut ja kertaviestit. Kuitatussa palvelussa jokainen vastaanottajasolmu palauttaa kuittausviestin lähettäjälle. Mikäli se ei vastaanota kuittauksia kaikilta kohdesolmuilta määräaikaan mennessä, lähettää se alkuperäisen viestin uudestaan. Lähettäjän odotusaika ja uudelleenlähetyksien lukumäärä ovat ohjelmoijan valittavissa. Toistavassa viestipalvelussa ei käytetä kuittausta, vaan perillemeno varmistetaan peräkkäisillä lähetyksillä. Näin vältetään kuittausviestien aiheuttama verkon kuormitus suurella vastaanottajamäärällä. Toistojen välinen aika ja lukumäärä ovat jälleen asetettavissa. Kertaviestipalvelussa ei tiedonsiirrolle ole varmistusmenetelmää, joten sovellus ei saa olla liian herkkä viestien katoamiselle. Toisaalta sillä saavutetaan suurin tiedonsiirtonopeus varsinkin suurilla datamäärillä, kun peräkkäisiä viestejä lähetettäessä ei tarvitse odottaa edellisen viestin siirtorutiinin päättymistä.

Neljäs tiedonsiirtopalvelu on protokollapinon yhteysjaksokerroksen tarjoama sovellusviestien kysely/vastauspalvelu (request/response). Palvelumuoto käyttää kuitatulle tiedonsiirtopalvelulle määriteltyä toistomäärää ja uudelleenlähetysaikaa protokollan ajastuksessa, nyt vain kuittausviesti sisältää lisäksi sovelluksen generoimaa dataa. Viestin vastaanottaneen solmun sovellusohjelma valmistelee vastauksen saamaansa viestiin ja lähettää vastauksen alkuperäiselle lähettäjälle. Palvelua käytettäessä tulee varmistaa sisääntulon sovelluspuskurien riittävä määrä sekä niiden vapautuminen saapuneiden viestien säännöllisellä käsittelyllä.

Verkkomuuttujan kyselypalvelussa (polling) verkkoa lukeva solmu pyytää sisääntuloverkkomuuttujansa päivitystä funktiolla ”poll( )”. Palvelumuoto muistuttaa sovellusviestien kysely/vastauspalvelua, nyt vain Neuron-piirin käyttöjärjestelmä huolehtii vastausviestin lähettämisestä ilman sovellusohjelman huomiota. Vastauksen generoivan solmun ulosmenevän verkkomuuttujan määre ”polled” estää verkkomuuttujan päivitysviestin välittömän lähetyksen, ja lähetys tapahtuu vasta päivityspyynnön saavuttua. Verkkomuuttujan uusi arvo ei ole välittömästi kysyvän solmun saatavilla, vaan sovellusohjelma sisältää tapahtumapohjaisen ajoitusrakenteen, jossa päivityksen saapuminen aktivoi uutta arvoa hyödyntävän koodin.

Kyselypalvelu edellyttää vastauksen generoivan solmun osoitetietojen kytkemistä kysyvän solmun sisääntuloverkkomuuttujaan, mikä vaatii yhden merkintäpaikan osoitetaulukosta. Kyselytoiminnon lisääminen järjestelmään muuttaa solmujen loogisessa yhdistämisessä käytettyä toimintamallia, ja verkonhallintaohjelmat tarvitsevat uudet versiot solmujen rajapintoja kuvaavista .xif-tiedostoista. Tämä on tärkeää huomata erityisesti kyselyn suorittavan solmun kohdalla, jossa kyselyn käyttöönotto ei vaikuta sisääntuloverkkomuuttujan määrittelyyn ohjelmakoodissa.

Tiedonsiirron palvelumuoto valitaan verkkoa suunniteltaessa ja tarkat määritykset tehdään asennuksen yhteydessä. Protokollan ajastimien asettamisessa huomioidaan verkon topologia, mikä määrää kulkuaikaviiveet ja heijastumat. Heijastumat aiheuttavat kohinaa ja viestien kahdentumista. Ennen uuden viestin lähettämistä on odotettava, että edellisen viestin lähetyksestä aiheutuneet heijastukset ovat vaimentuneet riittävästi. Vastaanotetun viestin tunnisteosaa säilytetään määrätty aika vastaanottotapahtumien puskurimuistissa, jotta kahdentuneet viestit voidaan tunnistaa.

3.5 Aidonnuspalvelu

Kuljetuskerros mahdollistaa aidonnuspalvelun, jossa lukeva solmu varmistuu kirjoittavan solmun oikeuksista. Lukeva solmu palauttaa kirjoittajalle 48-bittisellä aidonnusavaimella koodatun 64-bittisen satunnaisluvun, jonka kirjoittaja purkaa samalla avaimella ja lähettää lukevalle solmulle. Tämä suorittaa vielä kuittauksen, riippumatta salauksen purkamisen onnistumisesta.

Aidonnuspalvelun salausavaimen pituus on rajoitettu 48 bittiin johtuen USA:n ohjelmistojen vientiä koskevasta lainsäädännöstä. Salauksen murrettavuuden vuoksi eivät turva-alan yritykset kelpuuta sitä, vaan käyttävät omia valmistajakohtaisia ratkaisujaan (Piikkilä 2005). Aidonnuspalvelun käyttö lisää myös verkon kuormitusta tuntuvasti, kun yhden sovellustason viestin lähetys edellyttää neljän paketin siirtoa väylässä.

3.6 Sovellusviestit

Verkkomuuttujilla tapahtuvan yhteydenpidon lisäksi LonTalk-protokolla ja Neuron C -ohjelmointikieli tukevat sovellustason viestinvälitystä, jossa sovellusohjelma huolehtii viestien lähettämisen yksityiskohdista. Tällöin osa Neuron-piirin varusohjelmiston ja verkkoprosessorin automaattisista palveluista jää hyödyntämättä, mutta voidaan luoda oma räätälöity tiedonsiirron rajapinta, mikä mahdollistaa valmistajakohtaiset tiedonsiirtoratkaisut. Sovellusviestejä voidaan käyttää samanaikaisesti verkkomuuttujien rinnalla.

Sovellusviestien ainoa standardoitu käyttökohde on LonWorksin tiedostonsiirtoprotokolla. Kun verkkomuuttujan päivitysviestissä datakentän pituus on rajattu 31 tavuun, voidaan yhdessä sovellusviestissä siirtää 228 tavua dataa. Tämä mahdollistaa suurien datamäärien tehokkaan siirron, joskaan kaikki reitittimet eivät tue näin suuria paketteja.

3.7 Verkonhallintaviestit

Solmujen sovellusohjelmien välisen kommunikaation lisäksi LonTalk-protokolla mahdollistaa verkonhallintaviestit verkon ylläpitoon, solmujen asentamiseen ja niiden verkkoliikenteen asetusten muuttamiseen (Toshiba 2006). Tyypillisesti verkonhallintatyökalut lähettävät näitä viestejä, ja vastaanottavien solmujen varusohjelmisto reagoi niihin automaattisesti ilman sovellusohjelman väliintuloa. Sovellusohjelman kirjoittajan tulee ainoastaan huomioida mahdollisten verkonhallintaviestien vaikutus verkkoliikenteen puskurimuistien minimikokoihin.

4 NEURON C -OHJELMOINTI

Neuron C -ohjelmointikieli on suunniteltu Neuron-piiriä varten. Se pohjautuu ANSI-C:hen (ANSI 1989; Harbinson ym. 1991) ja sisältää laajennuksia hajautettujen ohjausjärjestelmien tarpeisiin. Laajennukset mahdollistavat viestinnän Lon-verkossa, Neuron-piirin I/O-toimintojen käytön sekä tapahtumapohjaisen ohjelmoinnin when-lauseen avulla.

Verkkomuuttujat, asetusparametrit sekä toiminnalliset lohkot määrittelevät LonWorks-laitteen verkkoliikenteen rajapinnan. Verkko- ja I/O-liikenteen toimintoihin on tarjolla valmiita funktioita ja tapahtumamäärittelyjä. Valmiiden ja itse määriteltyjen tapahtumien perusteella ohjataan tehtävien ajoitusta when-lauseita hyödyntäen.

4.1 Tapahtumapohjainen ajoitus

Neuron C tarjoaa sovellusohjelman kirjoittamiseen when-lauseen, jolla voidaan ajoittaa sovellusohjelman tehtävät tapahtumien perusteella. Tapahtuma voi olla ennalta määritelty, jolloin when-lauseen ehtona käytetään kääntäjään rakennettua avainsanaa. Valmiita tapahtumamäärittelyjä ovat esim. sisääntuloverkkomuuttujan arvon päivittyminen, lähtevän verkkomuuttujan päivityksen onnistuminen, muutos I/O-portissa ja ajastimen raukeaminen. Valmiiksi määriteltyjen tapahtumien käyttö on tehokasta ja se onnistuu vähäisellä koodimäärällä. When-lauseen ehtona voidaan lisäksi käyttää mitä tahansa Neuron C:n ilmaisua, joka voidaan tulkita loogisena tosi/epätosi-lausekkeena.

```
when (TAPAHTUMA TAI EHTO) {
        …TEHTÄVÄ…
}
```

Neuron-piirin käyttöjärjestelmä sisältää vuorottimen, joka ohjaa sovellusohjelman ajoa tapahtumatietojen perusteella. Vuorotin käy läpi when-lauseita kiertovuorotteluperiaatteella, ja when-lauseen ehtona olleen tapahtuman toteuduttua sen sisältämä ohjelmakoodi ajetaan. When-lauseille voidaan myös määritellä prioriteetti, jolloin vuorotin tarkastaa niiden ehdon toteutumisen ensimmäisenä jokaisella kierroksellaan. Mikäli jonkin prioriteettilauseen ehto on toteutunut, sen tehtävä suoritetaan ja vuorotin aloittaa jälleen alusta. Tapahtumatietojen välityksellä vuorotin tarjoaa sovellusohjelmalle tietoja verkkokerroksen tapahtumista, jotka ovat muuten piilossa sovellustasolta.

Neuron-piirin sovellusprosessori toteuttaa näennäisen rinnakkaisuuden when-lauseiden rajaamien ohjelmakoodin säikeiden välillä. Yksityiskohtaisemmin tarkasteltuna varusohjelmiston vuorotin siirtää ohjelman ajon ja sovellusprosessorin hallinnan when-lauseen sisältämälle tehtävälle. When-lauseen tehtävää suorittaessaan sovellusprosessori ei huomaa ohjelman ajoa määrittävien tapahtumatietojen muutoksia, vaan verkkoprosessori puskuroi tapahtumatiedot odottamaan sovellusprosessorin huomiota. Vasta when-lauseen sisältämän tehtäväkokonaisuuden suorituksen jälkeen sovellusprosessorin hallinta palautuu vuorottimelle, joka aloittaa tapahtumatietojen läpikäynnin valitakseen, mille prosessille se seuraavaksi luovuttaa sovellusprosessorin hallinnan. Sovellusohjelman kirjoittajalle tämä ohjelmointiympäristön ominaisuus asettaa rajoittavan tekijän, joka tulee huomioida työskentelyssä. Sovellusprosessorin hallinnan onnistunut palauttaminen vuorottimelle edellyttää ohjelmakoodin huolellista kirjoittamista, jotta esim. silmukoiden päättymisehdot toteutuvat.

4.1.1 Vahtikoira-ajastin

Neuron-piiri sisältää vahtikoira-ajastimen, jonka tehtävänä on resetoida piiri, mikäli sovellusohjelma ajautuu ikuiseen silmukkaan tai tapahtuu muu ajastimen asettamisen estävä ohjelmistovirhe. Resetointi tapahtuu, mikäli vahtikoira-ajastin ehtii raueta ja sitä ei ole ehditty liipaista uudestaan meneillään olevan jakson aikana. Raukeamisaika on kääntäen verrannollinen piirin kellotaajuuteen, ja 20 MHz piirille se on 0.42 s. Vahtikoira-ajastimen liipaisu tapahtuu asynkronisesti ajastimen laskemaan aikajaksoon nähden, jolloin resetointi tapahtuu 0.42 s ja 0.84 s välillä liipaisusta.

Normaalisti käyttöjärjestelmän vuorotin asettaa vahtikoira-ajastimen uudelleen sovellusprosessorin hallinnan palautuessa sille when-lauseen prosessilta. Prosessin pitkä kesto voi näin aiheuttaa piirin resetoitumisen. Tämä voidaan estää kutsumalla “watchdog_update( )”-funktiota pitkien tehtävien aikana. Funktiota ei tule käyttää silmukoissa, koska tämä estäisi laitteen toipumisen ikuisesta silmukasta. Myös sovellustason viestin vastaanottavat funktiot ”msg_receive( )” ja ”resp_receive( )” sekä tietyt I/O-funktiot liipaisevat vahtikoira-ajastimen.

4.1.2 Neuron-piirin haavoittuvuus sovellusohjelman ikuisille silmukoille

Neuron-piirin toteuttama ei-keskeyttävä tehtävien ajoitus ei kykene ottamaan sovellusprosessorin hallintaa prosessilta, joka haluaa vielä jatkaa ajoaan. Vahtikoira-ajastin varmistaa sovellusohjelmalle ulospääsyn ikuisesta silmukasta piirin resetoinnilla. Heti piirin käynnistymisen jälkeen muodostuva ikuinen silmukka voi kuitenkin jumiuttaa Neuron-piirin sovellusprosessorin lopullisesti. Vahtikoira-ajastin aiheuttaa tällöin piirin toistuvan ja nopeassa tahdissa tapahtuvan nollautumisen.

Peräkkäisten nollautumisten välisen ajan on oltava riittävän pitkä, jotta kohdepiiri ehtii reagoida tilanmuutoskäskyn sisältävään verkonhallintaviestiin ennen seuraavaa nollautumistaan. Neuron C -kääntäjä sisältää option ”Node recovery”, joka lisää resetoinnin yhteyteen viiveen. Tämä auttaa piiriä selviytymään toistuvista vahtikoira-ajastimen raukeamisen aiheuttamista nollautumisista.

NodeBuilderin laitemalli sisältää kääntäjän asetukset erikseen kehitys- ja julkaisuversiolle. Kehitysversio tukee debug-toimintoja, julkaisuversiosta sen sijaan karsitaan lopputuotteen kannalta tarpeettomat ominaisuudet ja sen kääntämisessä käytetään optimointialgoritmeja. Kehitysversiossa ”Node recovery” on oletuksena päällä, toisin kuin julkaisuversiossa. Toisaalta kehitysversion debug-toiminnot kasvattavat käännetyn binääritiedoston kokoa ja niitä voidaan hyödyntää vain LonMaker-verkonhallintaohjelmalla. Niinpä julkaisuversion käyttö myös sovelluskehityksessä on perusteltua, kunhan toiminto ”Node recovery” on kytketty päälle.

4.2 I/O-toiminnot

Neuron C:n sisäänrakennetut I/O-objektit helpottavat Neuron-piirin I/O-toimintojen käyttöä. Ne toimivat laiteajureina I/O-laitteistolle ja tarjoavat funktiokutsuihin pohjautuvan rajapinnan. Objektien arvoja hallitaan sovellusohjelmassa funktioilla ”io_in( )” ja ”io_out( )”. I/O-objekteja varten on myös valmiita tapahtumamäärittelyjä, joita voidaan käyttää tehtävien ajoitukseen.

Sovellusohjelmassa I/O-rajapinta toteutetaan määrittelemällä I/O-objekti, joka käyttää yhtä tai useampaa Neuron-piirin 11 I/O-pinnistä. Määrittelemätön pinni on oletuksena poiskytkettynä ja korkeaimpedanssisessa tilassa.

4.2.1 I/O-objektit

Neuron-piiri tarjoaa runsaan valikoiman vaihtoehtoja I/O-rajapinnan toteuttamiseksi. Vaihtoehtoina ovat suora, ajastimeen tai laskuriin pohjautuva, sarjamuotoinen sekä rinnakkaismuotoinen I/O-objekti, joista kaikista on tarjolla vaihtoehtoisia versioita.

Suoran I/O-objektin välityksellä tehdään välittömiä asetuksia I/O-pinnien arvoihin. Nämä objektit voivat muodostaa keskenään päällekkäisiä ryhmiä. Neuron-piiri sisältää kaksi ajastin-laskuria, joita käyttävät sisääntulon objektit voivat salvata sisään tulevaa signaalia piirin ajastimen ohjaamana. Vastaavaan ulostulon objektiin kirjoittaminen aiheuttaa reaktion vasta ajastetun jakson lopussa.

Neuron-piiri mahdollistaa samanaikaisesti vain yhden tyyppisen sarjaliikenteen I/O-objektin käytön. Sarjaliikenteen perustyypille sallitaan rinnakkaiset sisään- ja ulostulon objektit. Rinnakkainen objekti tarjoaa nopean kaksisuuntaisen tiedonsiirron, mutta varaa käyttöönsä kaikki piirin I/O-pinnit.

4.3 Ajastimet

Neuron C mahdollistaa ajastimien määrittelyn ja asettamisen muuttujien tapaan. Ajastimen raukeaminen voidaan asettaa tehtäviä ajoittavaksi tapahtumaksi. Ajastinmuuttuja tulee tällöin when-lauseen argumentiksi, ja when-lause ohjaa rajaamansa ohjelmakoodin suoritusta. Varusohjelmisto ilmoittaa sovellusohjelmalle ajastimen raukeamisen automaattisesti ja samalla käyttöjärjestelmän vuorotin siirtää ohjelman ajon ajastimen ohjaamien tehtävien suoritukseen.

```
when (timer_expires(my_timer)) {
        …TEHTÄVÄ…
}
```

Tarjolla on sekunnin ja millisekunnin ajastinobjektit, joita voi olla yhteensä 15 kappaletta. Objektien fyysisestä toteutuksesta vastaa Neuron-piirin kaksi 16-bittistä ajastinta, mikä rajaa sekuntiajastimen suurimmaksi arvoksi 65535 s. Millisekuntiajastimen suurin arvo on vain 32000 ms, kun Toshiban valmistamaa Neuron-piiriä ajetaan suurimmalla sallitulla 20MHz kellotaajuudella (Toshiba 2001).

Täsmällisimmät aikavälit voidaan muodostaa määrittelemällä ajastin toistuvaksi, jolloin ajastimen uudelleenalustusta ei tarvitse tehdä sovellusohjelmassa. Näin sovellusohjelman mahdollisesti hidas reagointi ajastimen raukeamiseen ei viivästytä sen uudelleenalustusta.

```
mtimer repeating my_timer; // toistuva millisekuntiajastin
```

Ohjelmointiopas sisältää laskukaavat ajastusarvon vaihteluvälille (Echelon 2003b). Kaavat huomioivat piirin kellotaajuuden, mikä vaikuttaa piirin sisään tulevan kellosignaalin erotuskykyyn.

4.4 Verkkomuuttuja

Verkkomuuttuja muodostaa LonWorks-sovelluksen perustan. Se toimii LonWorks-laitteen verkkoliikenteen rajapinnan peruselementtinä. Verkkomuuttujat määrittelevät laitteen sisään- ja ulostulot verkon näkökulmasta, ja ne mahdollistavat tiedon jakamisen hajautetussa järjestelmässä. Ulostuloverkkomuuttuja määritellään seuraavasti:

```
network output my_variable_type my_network_variable;
```

Sovellusohjelman kirjoittajalle verkkomuuttuja näkyy muuttujana, jonka arvo päivittyy automaattisesti laitteelta toiselle Lon-verkkoa pitkin. Kun sovellusohjelmassa kirjoitetaan ulostuloverkkomuuttujaan, saavat siihen kytketyt toisissa laitteissa olevat sisääntuloverkkomuuttujat päivitetyn arvon automaattisesti. Neuron-piirin varusohjelmisto huolehtii verkkomuuttujan arvon päivittymisestä muille laitteille, eikä sovellusohjelman kirjoittajan tarvitse huolehtia verkkoliikenteen yksityiskohdista. Sovellusohjelma ainoastaan varustaa ulostuloverkkomuuttujat viimeisimmillä arvoilla ja lukee muiden laitteiden voimassaolevat arvot sisätuloverkkomuuttujista.

Verkkomuuttujan uusi arvo etenee muihin solmuihin verkon tiedonsiirtonopeuden ja liikennetilanteen määräämässä ajassa. Ennen kuin tiedonsiirtokanavaan voidaan syöttää uutta viestiä, on odotettava toisen solmun mahdollisesti suorittaman lähetyksen päättymistä. Tämän edeltävän viestin pituuden ylärajana on eksplisiittisen viestin suurin mahdollinen 228 tavun datakentän pituus lisättynä tiedonsiirto-protokollan vaatimalla lisäinformaatiolla. Verkkoa muodostettaessa solmulle voidaan asettaa prioriteettiaikaikkuna, joka on voimassa tiettynä hetkenä jokaisen kanavassa lähetetyn viestin jälkeen. Prioriteettiaikaikkunaa käytettäessä viestin lähetys onnistuu suurella varmuudella heti edellisen lähetyksen päätyttyä. Eri solmujen prioriteettiaikaikkunat ovat peräkkäisessä järjestyksessä, jolloin ensimmäistä ikkunaa käyttävä solmu onnistuu tiedonsiirtokanavan varauksessa täysin varmasti ja myöhemmät aina, mikäli edeltäviä ikkunoita käyttävät solmut eivät ole aloittaneet prioriteettilähetystä.

Mikäli lähetyksessä ei käytetä prioriteettiaikaikkunaa, on viestin lähetyshetki voimakkaasti riippuvainen verkon kuormituksesta. LonTalk-protokollan liikennemääriin sopeutuva kilpavaraus estää törmäykset tehokkaasti suurellakin verkon kuormituksella, jolloin solmut arpovat lähetyshetkensä suuresta aikaikkunoiden joukosta. Tällöin verkkomuuttujan uuden arvon päivittymisnopeudesta voidaan tosin antaa vain arvioita, eikä ehdotonta maksimiaikaa voida määrittää. Tyypillisillä siirtonopeuksilla ja kohtuullisilla liikennemäärillä LonTalk-protokollan vasteaika on 10 - 20 ms (Piikkilä 2004).

Verkkomuuttujat helpottavat hajautetun järjestelmän luomista, koska yksittäiset laitteet voidaan kehittää itsenäisesti. Valmiita laitteita voidaan sitten helposti liittää toisiinsa erilaisten kokonaisuuksien muodostamiseksi.

4.4.1 Sovelluskerroksen verkkomuuttuja

Sovellusohjelman ja Lon-verkon näkemä verkkomuuttuja voi olla määriteltynä joko LonTalk-protokollan esitystapa- tai sovelluskerroksessa. Ylemmässä eli sovelluskerroksessa määritelty verkkomuuttuja sisältää esitystapakerroksen verkkomuuttujan ja lisäksi määrittelyt tiedon koodaukselle, skaalaukselle, vaihteluvälille, oletusarvolle ja yksikölle. Nämä määrittelyt on tehty sovelluskerroksen verkkomuuttujan resurssitiedostoon. Tarjolla on runsaasti standardeja LonMark-hyväksyttyjä verkkomuuttujia (SNVT), ja sovellusohjelman kirjoittaja voi myös luoda omia sovelluskerroksen verkkomuuttujia (UNVT). Helpoiten tämä voidaan tehdä NodeBuilder-ohjelmiston sisältämällä Resource Editor -ohjelmalla.

LonWorks-tekniikan kehittänyt Echelon suosittelee sovelluskerroksen verkkomuuttujien käyttöä, jotta laitteiden toteuttama verkkoliikenteen rajapinta olisi mahdollisimman monipuolisesti hallittavissa uusilla verkonhallintatyökaluilla. Sovelluskerroksen rakenteet dokumentoivat rajapinnan yksityiskohdat ja tarjoavat yhdenmukaisen menetelmän rajapinnan muodostamiseksi. Sovelluskerroksen verkkomuuttujiin lukeutuvat standardit verkkomuuttujat muodostavat perustan eri valmistajien välisen yhteentoimivuuden mahdollistaville avoimille ratkaisuille. Tämä on vallitseva käytäntö mm. rakennusautomaatiossa. Toisaalta sovelluskerroksen

määrittelyt voivat olla tarpeettomia ja raskaita valmistajakohtaisessa suljetussa järjestelmässä, josta tämän diplomityön ohessa valmistunut hissinohjausjärjestelmän simulointiympäristö toimii esimerkkinä.

4.5 Asetusparametri

Asetusparametrilla tehdään laitteeseen jokin perusasetus, esimerkiksi asetetaan verkkomuuttujan arvon keskipiste tai hystereesialue. Sillä säädetään yhtä tai useampaa verkkomuuttujaa tai toiminnallista lohkoa, tai laitetta itseään. Asetusparametri on verkkomuuttujan tavoin osana LonWorks-laitteen verkkoliikenteen rajapintaa. Se tarjoaa tarkasti määritellyn tavan tehdä laitteen asetukset verkon välityksellä.

Asetusparametri on LonTalk-protokollan ylimmän eli sovelluskerroksen rakenne. Sen koodaus, skaalaus, vaihteluväli, oletusarvo ja yksikkö määritellään resurssitiedostossa. Tarjolla on runsaasti standardeja LonMark-hyväksyttyjä asetusparametrien tyyppejä (SCPT), ja sovellusohjelman kirjoittaja voi myös luoda omia asetusparametrityyppejä (UCPT).

Asetusparametri toteutetaan joko konfiguroitavalla verkkomuuttujalla tai asetustiedostolla, jonka jälkeen se assosioidaan säätämäänsä kohteeseen. Tämä tehdään ominaisuuslistassa, joka sisältyy verkkomuuttujaa ja toiminnallista lohkoa säädettäessä niiden määrityslauseisiin. Kun parametrin halutaan suorittavan säätöä laitetasolla, sijoitetaan ominaisuuslista ohjelmakoodin tiedostotasolle itsenäiseksi rakenteeksi.

4.5.1 Konfiguroitava verkkomuuttuja

Konfiguroitava verkkomuuttuja käyttää verkkomuuttujaa asetusparametrin toteutukseen. Tällöin sen arvoa voidaan muuttaa verkonhallintatyökalulla sekä toisella LonWorks-laitteella, kuten muitakin verkkomuuttujia. Neuron C:n tapahtumapohjainen ajoitus kykenee reagoimaan konfiguroitavan verkkomuuttujan arvon päivittymiseen, jolloin päivittyminen voidaan helposti huomioida myös sovellusohjelmassa.

Konfiguroitavan verkkomuuttujan koko rajoittuu tavallisen verkkomuuttujan tapaan 31 tavuun. Itsenäisesti toimivassa Neuron-piirissä voi olla yhteensä korkeitaan 62 tavallista tai konfiguroitavaa verkkomuuttujaa. Konfiguroitava verkkomuuttuja määritetään sovellusohjelmassa verkkomuuttujan määreellä ”config_prop”.

4.5.2 Asetustiedosto

Asetusparametri voidaan toteuttaa konfiguroitavan verkkomuuttujan lisäksi asetustiedostolla, jolloin vältytään verkkomuuttujien koon ja lukumäärän rajoituksista. Kaikkien asetustiedostolla toteutettujen asetusparametrien tiedot kootaan samaan tiedostoon, joka on yhdistelmä eripituisia asetusmerkintöjä. Asetusparametrien tiedostoja voi olla myös kaksi, jolloin toinen on vain luettavia arvoja varten. Näiden asetustiedostojen on mahduttava kokonaisina kohdepiirin muistiin. Verkonhallintaohjelmaa varten voi olla lisäksi parametreja kuvaileva tekstitiedosto.

Asetustiedostolla toteutettuun asetusparametriin ei voida tehdä Lon-verkon loogisia kytkentöjä, jolloin asetusten teko on mahdollista ainoastaan verkonhallintatyökalulla tai siihen asennettavalla laitekohtaisella lisäohjelmalla. Sovellusohjelma ei myöskään havaitse asetustiedostoon tehtyä muutosta, ellei verkonhallintaohjelma suorita kohdepiirin resetointia, toiminnallisen lohkon poiskytkentää tai laitteen verkkoyhteyden poiskytkentää päivityksen yhteydessä.

Asetustiedostoon pohjautuva parametri toteutetaan määrittämällä ensin asetusperhe syntaksilla ”cp_family”. Asetusperhe on tyyppirakenne, joka sisältää tiedon käytetystä parametrin tyypistä, sen sallituista arvoista sekä parametrin muuttamiseen vaikuttavia asetuksia. Tyyppiä ” SCPTminSndT” edustavan asetusperheen ”cpMinSendT” määrittely:

```
SCPTminSndT cp_family cpMinSendT = { 0, 0, 1, 0, 0 }; // alkuarvot
```

Asetusperheen tunnusta käytetään säädettävän kohteen määrityksessä, ja tällöin asetusperhe saa uuden jäsenen eli tyyppirakenteensa mukaisen ilmentymän:

```
network output SNVT_lev_percent nvoValue
        nv_properties {
                cpMinSendT
};
```

Samaa asetusperhettä voidaan käyttää usean verkkomuuttujan, toiminnallisen lohkon ja laitteen asetusparametrin määritykseen, mikäli parametreille halutaan samanlaiset ominaisuudet. Asetusperheen määrittely ei vielä generoi ohjelmakoodia, vaan vasta sen tunnuksen käyttö ominaisuuslistassa luo parametrin fyysisen ilmentymän. Vasta tällöin parametria varten generoituu koodia ja sille varataan muistia. Konfiguroitavalle verkkomuuttujalle sen sijaan varataan fyysiset resurssit heti sen määrittelyn yhteydessä, ja ominaisuuslista luo tällöin ainoastaan loogisen kytkennän säädettävään kohteeseen.

Kun sovellusohjelmassa viitataan asetustiedostoon pohjautuvaan parametriin, tulee viittauksen sisältää kontekstitieto. Se yksilöi asetusperheen jäsenen mahdollisten useiden jäsenten joukosta. Kontekstitieto merkitään parametrin nimeä edeltävällä kontekstioperaattorilla ja parametrin suorittaman säädön kohteena olevan objektin nimellä. Kontekstioperaattorina toimii Neuron C:ssä kaksi peräkkäistä kaksoispistettä:

```
if (nvoValue::cpMinSendT.seconds > 0) { …
```

Koska samaan kohteeseen voidaan assosioida vain yksi tietyn tyyppinen asetusparametri, määrittelee tieto säädettävästä kohteesta asetusperheen jäsenen yksiselitteisesti. Laitekokonaisuutta suoraan säätävään asetusparametriin viitataan kontekstioperaattorilla, mutta säädön kohteen merkintä jää tällöin pois. Konfiguroitavaan verkkomuuttujaan voidaan viitata sovellusohjelmassa joko pelkällä verkkomuuttujan nimellä tai kontekstin perusteella.

Echelonin LonMaker-verkonhallintaohjelma tarjoaa graafisen esityksen verkkomuuttujista ja toiminnallisista lohkoista. Laitteen asetusparametreja säädetään tyypillisesti laitteen mukana toimitetulla lisäohjelmalla, joka asennetaan verkonhallintaohjelmaan. Lisäohjelma varustaa verkonhallintatyökalun räätälöidyillä asetusparametrien hallintakytkimillä. Suoritettaessa verkonhallintatoiminnot Echelonin kehittämän Lon-verkon käyttöjärjestelmän eli LNS:n kautta, tarjoaa LNS lisäohjelmille yhdenmukaisen rajapinnan. Tämä helpottaa laitetoimittajien työmäärää laitekohtaisten lisäohjelmien teossa, kun eri verkonhallintaohjelmille ei tarvitse tehdä erillisiä versioita lisäohjelmista.

4.6 Toiminnallinen lohko

Neuron C mahdollistaa verkkomuuttujien ja asetusparametrien ryhmittelyn tehtäväkokonaisuuksien mukaisiksi toiminnallisiksi lohkoiksi. Toiminnallinen lohko määritellään toiminnallisena profiilina, jonka lohko toteuttaa. Profiilin määrittely sisältää tietyn toiminnon vaatimat yhteiset rajapinnan komponentit. Näitä ovat pakolliset ja valinnaiset verkkomuuttujat sekä asetusparametrit. Profiilin toteuttava lohko sisältää kaikki pakolliset komponentit, halutun kokoonpanon valinnaisia komponentteja sekä lisäksi mahdollisia toteutuskohtaisia komponentteja.

Toiminnallinen lohko on LonTalk-protokollan sovelluskerroksen rakenne, ja toiminnalliset profiilit määritellään resurssitiedostoissa. Tarjolla on standardeja LonMark-hyväksyttyjä profiileja (SFPT), joihin voidaan haluttaessa tehdä lisäyksiä. Standardin profiilin mukaiseen toiminnalliseen lohkoon voidaan esim. lisätä oma verkkomuuttuja. Sovellusohjelman kirjoittaja voi myös luoda omia toiminnallisia profiileja (UFPT), jolloin verkkomuuttujakokoonpano voidaan määritellä vapaasti.

NodeBuilder-ohjelmisto käyttää standardeja resurssitiedostoja, jotka sisältävät LonMark-hyväksyttyjen verkkomuuttujien, asetusparametrien ja toiminnallisten profiilien määrittelyt. Omia profiilimäärittelyjä sisältäviä resurssitiedostoja voidaan luoda NodeBuilder-ohjelmiston sisältämällä Resource Editor -ohjelmalla. Laitevalmistajan määrittelemät omat käyttäjäkohtaiset verkkomuuttujat, asetusparametrit sekä toiminnalliset profiilit muistuttavat vastaavia standardeja rakenteita, ja ne voidaan tallettaa resurssitiedostoihin kuuluviin tyyppitiedostoon ja toimintaprofiilin asetustiedostoon. Resurssitiedostoihin kuuluvat lisäksi alustustiedosto verkkomuuttujille ja asetusparametreille sekä kieliasetuksia sisältävä tiedosto.

Neuron C -ohjelmakoodissa toiminnalliset lohkot määritetään fblock-määrittelyllä. Siinä sovelluksessa aiemmin määritellyt verkkomuuttujat assosioidaan profiilin sisältämiin verkkomuuttujiin. Seuraava lohko sisältää vain pakollisen verkkomuuttujan sekä suuntaajafunktion, josta kerrotaan luvussa 4.6.1:

```
network output SNVT_switch nvoSwitch;

fblock SFPTopenLoopSensor {
        nvoSwitch       implements nvoValue;
        director        SwitchDirector;
} Switch
```

Asetusparametrit assosioidaan toiminnalliseen lohkoon fblock-määrittelyn lopussa sijaitsevassa toiminnallisen lohkon ominaisuuslistassa. Kääntäjä toteuttaa fblock-määrittelyjen mukaiset assosiaatiot ja dokumentoi ne itsetunnistusdataan (SI), dokumentaatiodataan (SD) sekä .xif-tiedostoon. Nämä määrittelevät laiterajapinnan verkonhallintaohjelmalle. Tunnistus- ja dokumentaatiodata ovat osana käännetyn sovellusohjelman luettavaa tietorakennetta, ja ne välittävät verkonhallintaohjelmalle tiedot laitteen sisältämistä toiminnallisista profiileista, profiileihin assosioiduista komponenteista sekä rajapinnan komponenttien nimistä. Kääntäjän tuottama .xif-tiedosto toimitetaan yleensä laitteen mukana verkkoasennuksen helpottamiseksi, ja se sisältää yksityiskohtaiset tiedot laitteen toteuttamasta verkkoliikenteen rajapinnasta.

Toiminnallisen lohkon sisältämään tavalliseen ja konfiguroitavaan verkkomuuttujaan viitataan sovellusohjelmassa verkkomuuttujan nimen perusteella. Asetustiedostoon pohjautuvan parametrin viittauksen tulee sisältää kontekstitieto, joka yksilöi

asetusperheen jäsenen mahdollisten useiden jäsenten joukosta. Tämä tehdään parametrin nimeä edeltävällä kontekstioperaattorilla ja toiminnallisen lohkon nimellä.

Toiminnallisen lohkon tavallisia ja konfiguroitavia verkkomuuttujia voidaan myös käsitellä kontekstioperaattorin välityksellä. Tällöin viittauksessa käytetään verkkomuuttujille toiminnallisessa profiilissa määriteltyjä abstraktien jäsenten nimiä. Seuraavat lauseet toteuttavat saman toiminnon:

```
nvoSwitch.state = SWITCH_OFF; // viitataan verkkomuuttujan nimeen

Switch::nvoValue.state = SWITCH_OFF; // viitataan profiilin abstraktiin
                                     // jäseneen
```

Abstraktien jäsenten käyttö ohjelmakoodissa edistää modulaarista laitesuunnittelua ja ohjelmakoodin uudelleenkäyttöä. Profiilin tehtävää suorittava ohjelmakoodi voidaan siirtää eri ympäristöihin kun tehtävän tarvitsemat verkkomuuttujat voivat olla erinimisiä. Ne vain assosioidaan toiminnallisen lohkon käyttöön lohkon määrittelyssä.

Toiminnallisella lohkolla on myös tärkeä tehtävä eri valmistajien tuotteiden väliseen yhteentoimivuuteen pyrittäessä. Eri teollisuudenalat ja eturyhmät ovat vaikuttaneet standardien verkkomuuttujien valintaan, ja ryhmien toisistaan poikkeavien näkökulmien seurauksena LonMark-standardi sisältävää päällekkäisiä määritelmiä. Esim. lämpötila voidaan ilmaista kolmella eri SNVT:llä (Piikkilä 2004), jotka eivät ole keskenään yhteentoimivia. Toiminnallisten lohkojen määrittelyillä pyritään rajoittamaan variaatioiden määrää, joita on tarjolla laitteen verkkoliikenteen rajapinnan toteutukseen. Yhteentoimivuus toteutuu vasta, kun toiminnallisen kokonaisuuden käyttämä rajapinta on määritelty täsmällisesti.

4.6.1 Suuntaajafunktio

NodeBuilder-ohjelmisto mahdollistaa standardin LonMark-hyväksytyn verkkoliikenteen rajapinnan automaattisen muodostamisen sisältämällään Code Wizard -ohjelmalla. NodeBuilderin resurssitiedostoja hyödyntäen sillä voidaan generoida verkkoliikenteen rajapinnan toteuttava ohjelmarunko, johon sovellusohjelman kirjoittaja täydentää halutun toiminnallisuuden.

Automaattisesti Code Wizardilla generoitu ohjelmarunko perustuu suuntaajafunktioiden (director) käyttöön. Suuntaajafunktion avulla laite kykenee hallitsemaan toiminnallisia lohkojaan sekä ohjaamaan tapahtumat ja komennot oikeille toiminnallisille lohkoille.

Automaattisesti generoitu rajapinta koostuu toiminnallisista lohkoista, joita voidaan hallita NodeObject-lohkon välityksellä. Sen Request-sisääntulomuuttujan kautta voidaan verkonhallintatyökalulla pyytää valmista laitetta kytkemään, resetoimaan tai testaamaan minkä hyvänsä sisältämäänsä lohkon. NodeObject-lohko ohjaa verkosta vastaanottamansa lohkojenhallintakomennot eteenpäin kutsumalla säädön kohteena olevan lohkon suuntaajafunktiota. Ohjelman päätiedosto sisältää globaaleja when-lauseita, joista annetaan reagointikomentoja halutuille toiminnallisille lohkoille niiden suuntaajafunktioita kutsumalla.

Toiminnallinen lohko reagoi sovelluksen muista osista tuleviin herätteisiin sisältämänsä suuntaajafunktion kautta. Suuntaajafunktion argumenttina oleva komento toimii valitsimena funktion sisällä. Funktiota kutsuttaessa siitä suoritetaan else if -rakenteen ohjaamana komennon määrittämä osuus. Lohko voidaan muokata reagoimaan halutusti erilaisiin tapahtumiin lisäämällä ohjelmakoodia tapahtumaa edustavan komennon rajaamaan kohtaan.

Suuntaajafunktio ”myDirector” assosioidaan toiminnalliseen lohkoon sen määrittelyn loppuun lisättävällä lauseella:

```
director myDirector;
```

Lohkon sisällä määritellyn suuntaajafunktion argumentteina ovat lohkon indeksinumero sekä komento:

```
void myDirector (unsigned fbIndex, unsigned command);
```

Lohkojen suuntaajafunktiot on nimetty yksilöllisesti, mutta niitä kutsutaan yleisen director( )-funktion kautta. Tällöin kohteena olevaan lohkoon viitataan kontekstioperaattorin avulla, eikä lohkon indeksinumeroa tarvita. Seuraavassa kutsutaan lohkon ”Switch” suuntaajafunktiota:

```
Switch::director(UPDATE_COUNTER);
```

Suuntaajafunktion komentoja on määritelty yleisiä määrittelyjä sisältävän common.h-otsikkotiedoston luettelotyyppiin ”TFblock_command”. Valmiiksi määriteltyjen komentojen lisäksi voidaan tarpeen mukaisesti määritellä omia komentoja sovellusohjelman päätiedoston otsikkotiedostoon, josta määrittely näkyy kaikkialle:

```
#define UPDATE_COUNTER 17
```

Mikäli toiminnallinen lohko sisältää sisääntuloverkkomuuttujan, generoi Code Wizard kyseisen muuttujan päivittymiseen reagoivan when-lauseen. Ohjelmarunko kutsuu tästä when-lauseesta prosessifunktiota, johon sisääntuloverkkomuuttujan päivittymiseen reagoiva ohjelmakoodi tulee täydentää. Prosessifunktio on nimetty muotoon ”Lohkon_nimi_processNV( )”, esim. ”SwitchprocessNV( )”

Toiminnallisessa lohkossa voidaan viitata sen omaan verkkomuuttujaan normaalisti suoraan verkkomuuttujan nimellä. Lohkon toteuttaman verkkomuuttujan nimi voidaan tarkistaa lohkon .h-päätteisen otsaketiedoston määrittelyistä. Mikäli lohko sisältää useamman sisääntuloverkkomuuttujan, voidaan käyttää tiedostossa common.h määriteltyä rakennetta TDeviceState juuri päivittyneen verkkomuuttujan tunnistamiseen. Rakenne sisältää verkkomuuttujan globaalin indeksin, indeksin vektorimuuttujan päivittyneeseen elementtiin sekä toiminnallisen lohkon indeksin.

Tiedoston common.nc sisältämät funktiot käyttävät argumenttina toiminnallisen lohkon globaalia indeksiä. Siihen voidaan viitata kontekstioperaattorin avulla, seuraavassa viitataan lohkon ”Switch” indeksiin:

```
Switch::global_index
```

Lohkon indeksiin voidaan viitata myös funktion ”fblock_index_map( )” avulla, joka muuttaa verkkomuuttujan indeksin verkkomuuttujan sisältävän lohkon indeksiksi. Verkkomuuttujan indeksi saadaan verkkomuuttujasta funktiolla ”nv_table_index( )”:

```
fblock_index_map(nv_table_index(nviSwitch))
```

Suuntaajafunktioihin pohjautuvat toiminnalliset lohkot toteuttavat Code Wizardilla kehitetyn solmun verkkoliikenteen rajapinnan komponentit. NodeObject-lohkon sisääntuloverkkomuuttujaan lähetetyillä standardeilla pyynnöillä näitä lohkoja voidaan kytkeä päälle ja pois valmiissa laitteessa. Tämä ominaisuus mahdollistaa vaihtoehtoisten verkkoliikenteen rajapintojen toteuttamisen samaan laitteeseen,

jolloin toiminnallisia lohkoja voidaan kytkeä päälle käyttökohteen tarpeiden mukaisesti. Näin voidaan varmistaa eri valmistajien tuotteiden välistä yhteentoimivuutta.

4.7 Sovellusviestien lähetys ja vastaanotto

Verkkomuuttujilla tapahtuvan yhteydenpidon lisäksi LonTalk-protokolla ja Neuron C -ohjelmointikieli tukevat sovellustason viestinvälitystä, jossa lähetettävä viestipaketti muodostetaan ja lähetetään sovellusohjelmassa eksplisiittisesti. Viesti muodostetaan täyttämällä Neuron C:ssä valmiiksi määritellyn viestiobjektin ”msg_out” kentät ja viesti lähetetään kutsumalla funktiota ”msg_send( )”. Viestiobjekti ”msg_out” on määritelty seuraavasti:

```
struct {
        boolean priority_on;    // TRUE if a priority message (default:FALSE)
        msg_tag tag;            // message tag (required)
        int code;               // message code (required)
        int data[MAXDATA]; // message data (default:none)
        boolean authenticated; // TRUE if to be authenticated (default:FALSE)
        service_type service;   // service type (default:ACKD)
        msg_out_addr dest_addr; // see include file msg_addr.h (optional field)
} msg_out;
```

Ainoat pakolliset kentät viestiobjektissa ovat “tag” ja “code”. Muiden kenttien täyttäminen on vapaaehtoista.

Lähetettävään viestiin assosioidaan kohdeosoite erityisellä tunnuksella (tag). Kohdeosoitteen tunnus määrittelee viestien kohdeosoitteena käytettävän EEPROM-muistin osoitetaulukon merkinnän. Tunnukset kytkeytyvät loogisesti ensimmäisiin osoitetaulukon merkintöihin määrittelyjärjestyksensä mukaisesti. Kohdeosoitteen tunnus luodaan ohjelmakoodin alussa määrittelyllä:

```
msg_tag my_message_tag;
```

Lähetettävään viestiobjektiin määritelty kohdeosoitteen tunnus liitetään syntaksilla:

```
msg_out.tag = my_message_tag;
```

Viestiin liitetään numeerinen tunnuskoodi, joka määrittelee viestin sisällön vastaanottajalle. Itse määritellyn tunnuskoodin arvon on oltava väliltä 0…47. Muilla arvoilla tunnistetaan mm. verkkomuuttujien päivitysviestit ja verkonhallintaviestit. Viestin tunnuskoodi luodaan tyypillisesti ohjelmakoodin alussa esim. määrittelyllä:

```
#define MY_MESSAGE_CODE 12
```

Lähetettävään viestiobjektiin määritelty viestin tunnuskoodi liitetään syntaksilla:

```
msg_out.code = MY_MESSAGE_CODE;
```

Tunnuskoodin lisäksi sovellusviestissä voidaan lähettää muuta dataa kirjoittamalla se viestiobjektin kenttään ”data”, johon voidaan kirjoittaa haluttu määrä tavuja ulostulon sovelluspuskurin kapasiteetin rajoissa. Halutut arvot voidaan kirjoittaa kenttään tavu kerrallaan, jolloin käytetään syntaksia:

```
msg_out.data[0] = 7;
```

Data voidaan myös kopioida lohkona viestiobjektiin Neuron-piirin varusohjelmiston funktiolla ”memcpy( )” seuraavaa syntaksia käyttäen:

```
memcpy (msg_out.data, &my_struct, sizeof(my_struct));
```

“msg_out.data” on tässä kopioinnin kohteena. Lähteenä toimii jokin sovellusohjelmassa määritelty struct-rakenne. Kopiointia suoritetaan lähdeobjektin koon mukainen määrä, joka tarkastetaan varusohjelmiston funktiolla ”sizeof( )”.

Valinnaisesti täytettävässä kentässä “priority_on” voidaan viesti määritellä siirrettäväksi prioriteettipuskureiden välityksellä lähetin-vastaanottimelle ja lähetettäväksi tiedonsiirtokanavaan solmulle asetettua prioriteettiaikaikkunaa käyttäen. Kentässä ”authenticated” voidaan kytkeä päälle aidonnuspalvelu, jossa lukeva solmu varmistuu kirjoittavan solmun oikeuksista. Tällöin esim. kiinteistön lukituksesta huolehtiva järjestelmä ottaa vastaan viestejä vain solmuilta, jotka vastaavat oikein aidonnusavaimella koodattuun varmistusviestiin. Kenttään ”service” voidaan asettaa kuljetuskerroksen tarjoama tiedonsiirron palvelumuoto. Oletuksena käytetään kuittauspalvelua, jonka lisäksi voidaan käyttää kertaviesteihin pohjautuvaa, toistavaa sekä kysely/vastauspalvelua.

Sovellusviestin kohdeosoite voidaan muodostaa sovellusohjelmassa. Tällöin ei käytetä kohdeosoitteen tunnukseen eli ”tagiin” assosioitua EEPROM-muistin osoitetaulukon merkintää, vaan osoite muodostetaan eksplisiittisesti sovellusohjelmassa. Tämä tehdään täyttämällä osoitetiedot viestiobjektin muuttujaan ”dest_addr”. Tämä muuttuja on unioni, jonka alityyppi valitaan käyttöön muuttujan nimessä käytettävällä pisteoperaattorilla, esim. ”msg_out.dest_addr.snode.” Muuttujan ”dest_addr” tyyppimäärittely on unionin ”msg_out_addr” mukainen. Määrittely löytyy otsikkotiedostosta ”msg_addr.h”.

Viesti vastaanotetaan Neuron C:ssä valmiiksi määriteltyyn viestiobjektiin ”msg_in”. Saapunut viesti luetaan viestiobjektin kentistä, jotka on määritelty seuraavasti:

```
struct {
        int code;                   // message code
        int len;                    // length of message data
        int data[MAXDATA];          // message data
        boolean authenticated;      // TRUE if message was authenticated
        service_type service;       // service type used by sender
        msg_in_addr addr;           // see <msg_addr.h> include file
        boolean duplicate;          // the message is a duplicate
        unsigned rcvtx;             // the message's receive tx ID
} msg_in;
```

Kenttä “code” sisältää lähettäjän antaman tunnuskoodin, joka määrittelee viestin sisällön. Kenttä ”len” kertoo tavumäärän, jonka viestin lähettäjä on tarkoittanut luettavaksi data-kentästä. Data-kentän arvot luetaan sisääntulon sovelluspuskurista, jossa on merkityksellistä dataa vain kentän ”len” ilmoittama tavumäärä. Kentän ”data” arvot voidaan lukea tavu kerrallaan, jolloin käytetään syntaksia:

```
my_variable = msg_in.data[0];
```

Data voidaan myös kopioida lohkona viestiobjektista Neuron-piirin varusohjelmiston funktiolla ”memcpy( )” seuraavaa syntaksia käyttäen:

```
memcpy(&my_struct, msg_in.data, sizeof(my_struct));
```

Kopioinnin kohteena toimii nyt sovellusohjelmassa määritelty struct-rakenne. Kohteena voi lisäksi olla vektori, vektorin elementti tai osoittimen mukainen muu muistialue. Tätä rakennetta käytettäessä tulee olla erityisen huolellinen, jottei kirjoitus ohjaudu väärään paikkaan!

Ennen kopiointia tulee tarkastaa, onko vastaanotetulla viestillä oikea pituus. Näin selvitetään etukäteen, ovatko edellytykset kopioimiselle olemassa. Edellisessä ”memcpy( )”-esimerkissä käytetään ehtoa:

```
if (msg_in.len == sizeof(my_struct)) { // tarkastetaan viestin koko
```

Viestiobjektin kenttä ”authenticated” ilmoittaa aidonnuspalvelun käytön ja ”service” käytetyn tiedonsiirron palvelumuodon. Viestiin merkityt lähde- ja kohdeosoitteet voidaan tarkistaa kentästä ”addr”, jonka tyyppi “msg_in_addr” on määritelty otsikkotiedostoon ”msg_addr.h”.

Kenttä ”duplicate” ilmoittaa tiedonsiirron kysely/vastauspalvelua käytettäessä, onko viesti aiemmin vastaanotetun kahdentuma. Kysely/vastauspalvelussa kyselyn vastaanottaneen solmun sovellusohjelma valmistelee vastauksen saamaansa viestiin ja lähettää vastauksen kysyjälle. Vastaanotettaessa toistettu kysely voidaan vastausdatan uudelleen generoiminen välttää, mikäli jo generoidut vastaukset puskuroidaan vastaanottotapahtuman indeksin perusteella ja toistettuihin kyselyihin vastataan puskuroiduilla vastauksilla. Saapuneen viestin vastaanottotapahtuman indeksi voidaan lukea kentästä ”rcvtx”.

Sovellusviestien saapumisen havaitsemiseksi Neuron C sisältää valmiin tapahtumamäärittelyn ”msg_arrives( )”, johon voidaan liittää tarkennukseksi viestin tunnuskoodi. Tällä tapahtumatiedolla ohjataan saapuvan viestin käsittelevää when-lausetta:

```
when (msg_arrives(MY_MESSAGE_CODE)) {
```

Tunnuskoodin sisältävän ehdollisen saapumistapahtuman lisäksi tulee myös tarkastaa yleinen saapumistapahtuma, joka ei sisällä viestin tunnuskoodia. Tämä tulee tehdä, jotta mahdollisesti saapunutta tunnistamatonta viestiä varastoiva verkkoliikenteen puskurimuisti vapautuu. Yleinen muoto reagoi kaikkiin saapuviin viesteihin ja kuittaa myös niiden saapumistiedot, jolloin puskurimuisti vapautuu. Tunnistamattomia viestejä ei tarvitse käsitellä sen enempää, ainoastaan viestien saapumisen tapahtumatieto kuitataan seuraavasti:

```
when (msg_arrives) {
        ;
}
```

Viestiä ja sen saapumistietoa säilytetään sisääntulon sovelluspuskurissa. Puskurin sisältö muodostaa jonon, jota on pakko käsitellä järjestyksessä. Mikäli sovellusohjelma ei käsittele jonon kärjessä olevaa tapahtumatietoa, jonon käsittely pysähtyy eikä sovelluspuskureita voida vapauttaa. Tällöin tapahtumien normaali käsittely estyy eikä laite kykene reagoimaan myöhemmin saapuneisiin viesteihin. Sovellusviestejä käytettäessä tulee sovellusohjelman aina huomioida saapuneet viestit säännöllisesti sisääntulopuskureiden vapauttamiseksi, koska ne eivät vapaudu ilman sovellusohjelman huomiota automaattisesti kuten verkkomuuttujia käytettäessä.

Uuden viestin saapuessa on ensin suoritettava ehdollisen saapumistapahtuman tarkistava when-lause ja vasta sen jälkeen yleinen. Muuten yleisen saapumistapahtuman tarkastus tyhjentää viestin sovelluspuskurista ennen aikojaan. Viesti voi saapua juuri ennen tehtävän suorituksen siirtymistä yleiselle when-lauseelle. Tällöin when-lauseet suoritetaan väärässä järjestyksessä, ellei Neuron-piirin käyttöjärjestelmässä tehtävien ajoitusta ohjaavan vuorottimen nollaus ole päällä. Tällöin tehtävien suoritus alkaa aina sovelluksen alusta ajastimen rauetessa tai viestin saapuessa. Nollaus asetetaan kääntäjän käskyllä:

```
#pragma scheduler_reset
```

Vuorottimen nollausta käytettäessä viestejä ja niiden tapahtumatietoja säilövät sovelluspuskurit ovat edelleen vaarassa tukkeutua. Esim. toistuvasti umpeutuvan ajastimen aiheuttama vuorottimen tiheä nollaus voi aiheuttaa viestiliikenteen tukkeutumisen, mikäli ohjelmakoodin lopussa esiintyvää when-lausetta ei ehditä suorittaa. Tämän estämiseksi harvoin ajettavat when-lauseet sijoitetaan ohjelmakoodin alkuun. Loppuun sijoitetaan puolestaan ne when-lauseet, joiden liipaisuehto toteutuu usein. Sovelluksen viimeinen when-lause suoritetaan vasta muiden tehtävien jälkeen, vaikka sen liipaisuehto olisi jo toteutunut.

Mikäli samassa tehtävässä sekä vastaanotetaan että lähetetään sovellusviesti, tulee tarpeelliset vastaanotetun viestiobjektin tiedot kopioida talteen ennen uuden viestin muodostuksen aloitusta. Msg_in-kentän tiedot saatetaan menettää arvoja msg_out-objektiin sijoitettaessa. Saapunut viesti menetetään myös tehtävän suorituksen päättyessä, eli toisesta when-lauseesta ei voida viitata msg_in-objektiin, jonka sisältö vastaanotettiin edellisessä when-lauseessa.

4.8 Neuron C:n poikkeavuuksia totutusta C-ohjelmoinnista

Neuron-piirin prosessorit ovat kahdeksanbittisiä, joten sovellusohjelmakin ajetaan tehokkaimmin kahdeksanbittisiä muuttujia käyttäen. Tällaisen muuttujan suurin arvo on etumerkillä varustettuna 127 ja ilman etumerkkiä 255. PC:lle C-ohjelmia kirjoittanut joutuu opettelemaan, että Neuron C:ssä muuttujat ovat oletuksena kahdeksanbittisiä, ja määreellä “long” bittimääräksi tulee 16.

Neuron C -sovellusohjelma ei sisällä ”main( )”-funktiota. Sen sijaan sovellusohjelman varsinainen ajettava koodi sijaitsee funktioissa sekä when-lauseissa. When-lause toteuttaa yhden ohjelmakoodin säikeen, joita sovellusprosessori suorittaa näennäisesti rinnakkaisina vuorottimen ohjaamana.

Erikseen käännettäviä lähdekooditiedostoja ei tueta, mutta ”#include”-käskyä voidaan käyttää otsaketiedostojen liittämiseen.

Neuron C -kääntäjä hallitsee seuraavien otsaketiedostojen sisältämät ANSI-C:n kirjastofunktioiden määrittelyt: stddef.h, stdlib.h ja limits.h. Sisällytettyinä ovat myös merkkijono- ja bittioperaatiot standarditiedostosta string.h. Tarjolla on lisäksi 28 muuta otsaketiedostoa, jotka sisältävät Neuron-sovelluksiin räätälöityjä funktioita (Echelon 2003a).

Varusohjelmisto alustaa oletuksena globaalit muuttujat automaattisesti nolliksi piirin käynnistyksen ja resetoinnin yhteydessä. Tämä koskee myös staattisia muuttujia, I/O-objekteja, sisääntuloverkkomuuttujia sekä ajastimia. Toiminnon aktiivisella hyödyntämisellä voidaan pienentää sovellusohjelman tilantarvetta. Ainoastaan funktioiden sisäisiä paikallisia verkkomuuttujia ei alusteta, mikäli niitä ei ole määritelty staattisiksi.

Neuron C:n syntaksi sisältää kontekstioperaattorin, jona toimii kaksi peräkkäistä kaksoispistettä. Se yksilöi asetusperheen jäsenen mahdollisten useiden jäsenten joukosta, kun sovellusohjelmassa viitataan asetustiedostoon pohjautuvaan parametriin. Kontekstitieto merkitään parametrin nimeä edeltävällä kontekstioperaattorilla ja parametrin suorittaman säädön kohteena olevan objektin nimellä.

Neuron C:ssä ei struct-rakennetta eikä unionia voida käyttää proseduurin parametrina tai funktion paluuarvona. Verkkomuuttujan tai asetusparametrin struct-rakenne ei voi sisältää osoittimia.

4.8.1 Muistien lähialueet

Sovellusohjelman linkityksen yhteydessä tulee usein virheilmoitus, että ohjelma ei mahdu käytettävissä olevaan muistiin. Tämän aiheuttaa oletuksena käytettävä lähialueen osoite, jolla voidaan osoittaa vain 256 tavua.

Oletuksena Neuron C:ssä määritellylle muuttujalle varataan tilaa RAM-muistin lähialueesta. Myös EEPROM-muuttujan oletussijainti on EEPROM-muistin lähialueessa. Neuron-piirin muistiavaruutta osoitetaan muuten 16 bitillä, mutta muistien lähialueiden osoittamiseen riittää kahdeksan bittiä. Tämä mahdollistaa lyhyemmät käskyt ja nopeammat muistitoiminnot, mikäli sovellusohjelma lukee tai kirjoittaa muistiin suoraan. Epäsuora osoittimen välityksellä tapahtuva muistin käyttö ei hyödy lähialueen osoitemuodosta.

Muuttujan tarvitsema muistitila voidaan varata lähialueen ulkopuolelta, kun määrittelyyn lisätään määre ”far”. Tämä kannattaa tehdä osoittimen välityksellä tavoitettaville tietorakenteille jo sovelluskehityksen alkuvaiheessa. Näin lähialue voidaan hyödyntää tehokkaasti usein käytettävillä muuttujilla, joita hallitaan suoraan ilman osoitinta.

4.8.2 EEPROM- ja flash-muistiin kirjoittaminen osoittimen välityksellä

Sovellusohjelma sijaitsee Neuron-piirin integroidussa EEPROM-muistissa. Se sisältää verkkoyhteyden fyysisen tason toiminnan mahdollistavat kokoonpanon parametrit sekä verkkoasetustaulukoiden ja puskurimuistien muistialueita määrittävät sovelluskohtaiset parametrit. EEPROM-muisti sisältää aina myös verkkoliikenteen asetustaulukot. Mikäli Neuron-piiriin on kytketty ulkoista flash-muistia, sisältää se varusohjelmiston sekä mahdollisesti myös sovellusohjelman.

Koska Neuron-piirin integroitu EEPROM-muisti ja mahdollinen ulkoinen flash-muisti sisältävät tärkeitä asetustietoja, on Neuron C:ssä oletuksena estetty näille muistialueille kirjoittaminen osoittimen välityksellä. Neuron C -kääntäjä tulkitsee EEPROM- tai flash-muistiin osoittavan osoittimen vakion osoittimeksi. EEPROM-muuttujan lisäksi myös verkkomuuttujan ja asetusparametrin osoitinta kohdellaan vakion osoittimena, mikä estää muutosten tekemisen osoittimen välityksellä. Tähän voidaan tehdä poikkeus kääntäjän käskyllä ”#pragma relaxed_casting_on”, mikä kuitenkin ohittaa kääntäjän varmistusmekanismit ja mahdollistaa myös virheellisen kirjoituksen EEPROM- tai flash-muistiin.

4.9 Neuron-piirin toimintatilat

Ohjelmointityökalulla kirjoitettu ja käännetty Neuron C -ohjelmakoodi siirretään ohjelmointiin käytetyltä PC:ltä kohdepiiriin verkonhallintatyökalulla, joka kytkeytyy Lon-verkkoon verkkosovittimen välityksellä. Ohjelmakoodin latauksen aikana ja sen jälkeen Neuron-piirin toimintatilaa voidaan tarkkailla sen vieressä piirilevyllä sijaitsevan huolto-LEDin perusteella.

Taulukko 2. Neuron-piirin toimintatiloja vastaavat huolto-LEDin indikaatiot.

Solmun toimintatila	Huolto-LED
sovellukseton ja asetukseton	päällä jatkuvasti
asetukseton, sisältää sovelluksen	vilkkuu
yhteydetön tila, sisältää asetukset	pois päältä
normaali tila, sisältää asetukset	pois päältä

Neuron-piirillä on neljä toimintatilaa, jotka säilyvät resetoinnin yhteydessä. Lisäksi normaalissa tilassa oleva piiri voidaan asettaa tilapäisesti yhteydettömäksi (soft off-line), jolloin tämä asetus ei säily piirin resetoinnissa. Tietoa piirin tilasta ylläpidetään EEPROM-muistissa ja se asetetaan verkonhallintaviestillä ”set node mode”. Tällä viestillä voidaan muuttaa piirin tilaa, muuttaa tilapäistä yhteystilaa sekä resetoida piiri. Piirin tilaa voidaan tiedustella verkonhallintaviestillä ”query status”.

Piiri on sovellukseton ja asetukseton, kun sovellusta ei ole vielä ladattu piiriin, sovelluksen lataus on kesken tai sovelluksen lataus on tarkistussumman perusteella epäonnistunut. Tällöin Neuron-piirin huolto-LED palaa jatkuvasti, eikä sovellusta voida ajaa. Mikäli sovelluksen lataus epäonnistuu, voidaan piiri resetoida ja yrittää latausta uudelleen.

Asetuksettomassa tilassa huolto-LED vilkkuu sekunnin välein. Tällöin Neuron-piiri sisältää sovelluksen, mutta asetukset ovat vielä tekemättä, niiden lataus on meneillään tai ne on tarkistussumman perusteella asetettu väärin. Yhteydettömässä tilassa Neuron-piiri sisältää sovellusohjelman ja asetukset. Sovellusohjelmaa ei kuitenkaan ajeta, sillä käyttöjärjestelmän vuorotin on kytketty pois. Huolto-LED ei tällöin pala.

Ainoastaan normaalissa toimintatilassa sovellusviestit vastaanotetaan, koska ne vaativat sovellusohjelman käsittelyä. Muissa pysyvissä tiloissa Neuron-piiri ei reagoi verkkomuuttujien kyselyihin eikä päivitä sisääntuloverkkomuuttujia. Soft off-line -tilassa sisään tulevat verkkomuuttujien päivitykset käsitellään, mutta vuorotin ei reagoi päivityksen luomaan tapahtumatietoon. Verkkomuuttujan kysely palauttaa tällöin tyhjän paketin.

Verkonhallintaa suoritettaessa on suositeltavaa asettaa kohteena oleva laite yhteydettömään tilaan ennen muiden verkonhallintaviestien lähettämistä, koska niiden käsittely aiheuttaa vääristymiä järjestelmän sovellusohjelmalle tarjoamiin ajastinpalveluihin. Käytettäessä verkkomuuttujan määrittelyssä määrettä ”bind_info(offline)” tai asetusparametrille määrettä ”cp_info(offline)”, asettaa verkonhallintaohjelma laitteen yhteydettömään tilaan aina ennen kuin se tekee muutoksia kyseiseen muuttujaan tai parametriin. Tätä asetusta käyttämällä sovellusohjelma kykenee havaitsemaan asetustiedostolla toteutetun asetusparametrin päivityksen, joka ei muuten generoi tapahtumatietoa Neuron-piirin vuorottimelle.

Neuron C sisältää valmiit tehtävien ajoitusta ohjaavat tapahtumamäärittelyt ”offline” ja ”online”. Näiden määrittelyjen rajaamat ohjelmakoodit ajetaan kun vastaava verkonhallintaviesti vastaanotetaan, ja vasta tämän jälkeen piirin toimintatila vaihtuu.

Näin sovellus voi valmistautua uuteen toimintatilaan esim. kytkemällä oheislaitteita tai pysäyttämällä ajastimet. Tapahtumaa “offline" voidaan käyttää myös hätätapauksissa, ylläpidossa tai reagointina johonkin järjestelmän laajuiseen tilaan.

4.9.1 Binäärikoodin latautumisnopeus Neuron-piirin EEPROM-muistiin

Ohjelmointityökalun linkittäjä sisältää mallin kohdepiirin ja mahdollisen ulkoisen muistin tarjoamista fyysisistä resursseista. Se ratkaisee globaaleihin symbolisiin osoitteisiin tehdyt viittaukset, ohjaa sovelluskoodin oikeille muistialueille sekä varaa muistialueet sovellusohjelman käyttöön. Fyysisen muistin varaus loogisten rakenteiden tarpeisiin tapahtuu näiden määrittelyjärjestyksessä. Mikäli sovellusohjelmassa määriteltyjen objektien järjestys pysyy samana, voivat myös linkitetyt osoitteet säilyä samoina ohjelmaa uudelleen käännettäessä (Echelon 2003b).

Toshiban valmistaman Neuron-piirin EEPROM-muistin jokaiseen tavuun voidaan kirjoittaa 10 000 kertaa (Toshiba 2006; Echelon 2004). Yhden tavun kirjoittamiseen kuluu 20 ms. Kirjoitustapahtuman nopeuttamiseksi ja EEPROM-muistin säästämiseksi varusohjelma vertaa kirjoitettavaa arvoa muistin jo sisältämään arvoon. Mikäli nämä täsmäävät, ei kirjoittamista suoriteta (Toshiba 2006).

Mikäli sovellusohjelman koodiin tehdään vain pieni muutos eikä ohjelmakoodin osien järjestystä muuteta, latautuu uusi binäärikoodi Neuron-piiriin hyvin nopeasti. Jos taas ohjelman osien järjestystä muutetaan, joudutaan muistin sisältöä muuttamaan laajemmin ja binäärikoodin latautuminen kestää tuntuvasti kauemmin.

5 NEURON-PIIRIN TIETORAKENTEET

Neuron-piirin ohjelmisto koostuu varus-, sovellus- ja verkko-osioista. Ne toteuttavat piirin käyttöjärjestelmän, verkkosolmun paikallisen tehtävän Lon-verkossa sekä varustavat solmun verkkoliikenneasetuksilla. Sovellusohjelman kirjoittajan on hyvä tuntea ohjelmiston eri osien tehtävät, jotta hän osaa hyödyntää sovellusohjelman kirjoittamiseen tarjolla olevia palveluja, hallita Neuron-piirin niukkoja muistiresursseja, optimoida verkon suorituskykyä sekä tehdä verkkoliikenteen vaatimia asetuksia.

Verkkoliikenteen asetusten tekeminen niille varattuihin asetustaulukoihin muodostaa oman aihekokonaisuutensa, jota tarkastellaan yksityiskohtaisesti seuraavassa luvussa. Aiheella on erityinen painoarvo tässä diplomityössä, koska työn ohessa valmistuneessa hissinohjausjärjestelmän simulointiympäristössä verkkoasetukset asetetaan Neuron-piirin sovellusohjelmasta käsin.

5.1 Varusohjelmisto

Neuron-piirin varusohjelmisto sisältää LonTalk-protokollan, tapahtumatietojen pohjalta tehtäviä ajoittavan vuorottimen sekä ajonaikaiset kirjastofunktiot. Käyttöjärjestelmätehtävien ohella varusohjelmisto pyrkii helpottamaan sovellusohjelman kirjoittamista Neuron-piirille. Lon-verkon solmujen välinen tietoliikenne on helppo toteuttaa valmiiden funktioiden ja vuorottimen tarjoamien palvelujen avulla.

Varusohjelmiston uudemmat versiot vaativat 16Kb:n tilan, mikä on paljon verrattuna sovellusohjelmalle tyypillisesti varattuun 3Kb:n EEPROM-muistiin ja 4Kb:n RAM-muistiin (Toshiba 2001). Neuron 3120 -piirissä varusohjelmisto sijaitsee integroidussa ROM-muistissa ja ulkoista muistia sisältävässä 3150-piirissä ulkoisessa ROM- tai flash-muistissa, jonne se voidaan kirjoittaa esim. NodeBuilder-ohjelmointityökalulla.

5.1.1 Ajonaikaiset kirjastofunktiot

Neuron C tarjoaa ohjelmoijan käyttöön joukon funktioita ohjelman suorituksen hallintaan, verkkoasetusten tekoon, lasku- ja merkkijono-operaatioihin sekä I/O-toimintoihin. Neuron C -kääntäjä tuntee nämä funktiot ja ne ovat osana Neuron-piirin varusohjelmistoa. Kaikki funktiot eivät tosin ole mahtuneet piirin varusohjelmistoon, ja näitä funktioita käytettäessä ohjelmointityökalu linkittää ne osaksi sovellusohjelmaa sen kääntämisen yhteydessä. Tällöin sovellusohjelman koko kasvaa, mikä verottaa niukkaa EEPROM-muistia. Sovellusohjelman kirjoittajan onkin hyvä tarkistaa kohdepiirinsä datalehdestä, mitkä funktiot löytyvät valmiiksi piirin varusohjelmistosta ja pyrkiä ensisijaisesti niiden käyttöön. Varusohjelmiston koko on jouduttu rajoittamaan integroidun ROM-muistin kapasiteetin mukaiseksi, mutta vuosien myötä tämä kapasiteetti on kasvanut.

Ulkoista muistia käyttävä Neuron-piirin versio ei sisällä lainkaan integroitua ROM-muistia, vaan varusohjelmisto sijaitsee ulkoisessa ROM- tai flash-muistissa. Tällöin varusohjelmiston kokorajoitukset poistuvat, ja ohjelmiston standardiversioon voidaan linkittää lisää toimintoja. Laitevalmistaja voi lisätä varusohjelmistoon funktioita, jotka helpottavat esim. tuotteeseen räätälöidyn I/O-laitteiston käyttöä. Näin helpotetaan myöhempää sovellusohjelmien kääntämistä laitteelle. Toimintojen sijoittaminen räätälöityyn varusohjelmistoon nopeuttaa myös sovellusohjelman kääntämistä, koska tällöin varusohjelman sisältävää osaa ei tarvitse kääntää. Tekniikka tukee myös modulaarista ohjelmointimallia, jossa ohjelmakoodin uudelleenkäytöllä parannetaan luotettavuutta ja lyhennetään kehitystyöhön kuluvaa aikaa.

5.2 Sovellusohjelma

Sovellusohjelma toteuttaa solmun paikallisen tehtävän, esim. mittauksen tai toimilaitteen säädön. Ohjelmakoodi ajetaan Neuron-piirin ylimmällä tasolla eli sovellusprosessorissa, jolle siirtotie- ja verkkoprosessorissa tuotetaan palveluja. Nämä sovellusohjelma tavoittaa varusohjelman funktioiden ja vuorottimen toimintojen välityksellä.

Sovellusohjelma kirjoitetaan kehitystyöhön käytetyn tietokoneen ohjelmointityökalulla, joka kääntää ohjelmakoodin ja linkittää ohjelman vaatiman muistin tarjolla olevaan fyysiseen muistiin. Tämä edellyttää, että ohjelmointityökalu sisältää mallin kohdepiiristä ja mahdollisista ulkoisista muistipiireistä. Käännetty sovelluskoodi on myös riippuvainen Neuron-piirin sisältämästä varusohjelmistosta, ja sovellusohjelman kutsumat palvelut linkitetään varusohjelmistoon.

Mikäli sovellusohjelma ladataan Neuron-piirin integroituun EEPROM-muistiin, voidaan siirto tehdä Lon-verkkoa pitkin verkonhallintatyökalulla, joka hyödyntää LonTalk-protokollan verkonhallintakäskyjä. Ulkoista muistia käyttävässä Neuron-piirissä sovellusohjelma sijaitsee yleensä ulkoisessa ROM- tai flash-muistissa, jos sen kapasiteetti on riittävä. Ulkoiselle muistipiirille sijoitetaan kuitenkin ensisijaisesti varusohjelmisto, joka ei voi sijaita integroidussa EEPROM-muistissa.

5.2.1 Kokoonpanon parametrit

Sovellusohjelma sisältää asetukset Neuron-piirin kellotaajuudelle, lähetin-vastaanottimen tyypille sekä verkon bittinopeudelle. Nämä tiedot mahdollistavat yhteyden Neuron-piiristä Lon-verkkoon, kun Neuronin varusohjelmisto saa tiedon siihen kytketyn lähetin-vastaanottimen liityntäpinnasta. Tiedot ovat osana EEPROM-muistissa sijaitsevia kokoonpanon parametreja.

Loput kokoonpanon parametreista lukeutuvat verkkoasetuksiin ja ne asetetaan yleensä solmua verkkoon asennettaessa. Lähetin-vastaanottimen toimintaa ohjataan mm. törmäyksen tunnistuksen parametreilla ja viidellä siirtotieprosessorin ajastusparametrilla. Lisäksi asetetaan solmulle siirtotien tarjoavan kanavan tunnus ja prioriteettiaikavälien lukumäärä, solmun prioriteettinumero, solmun fyysistä sijaintia verkonhallintaohjelmalle kuvaileva merkkijono, verkonhallintaviestien aidonnus, sovellusohjelman estotilan ajastin sekä vastaanottoajastin muille kuin ryhmä- ja verkonhallintaviesteille. Parametreista on tarjolla yksityiskohtainen kuvaus (Toshiba 2006).

Neuron-piirin verkkoasetukset asetetaan tyypillisesti verkonhallintaohjelmalla Lon-verkkoa pitkin. Tällöin kuitenkin kommunikoinnin perusasetusten on jo oltava kohdallaan verkkoyhteyden mahdollistamiseksi. Neuron-piirin varusohjelmiston on tiedettävä, minkä tyyppisen lähetin-vastaanottimen kautta se kommunikoi. Niinpä nämä fyysisen siirtotien mahdollistavat parametrit on sijoitettu sovellusohjelmaan.

Sovellusohjelman kehitys tapahtuu tyypillisesti Lon-verkon laitteita valmistavassa yrityksessä. Tuotteina ovat esim. lämpötila-, paine- ja virtausanturit sekä verkon kautta tapahtuvaan säätöön reagoivat venttiilit, moottorit, lämmittimet ja pumput (Echelon 1995b). Valmiit laitteet myydään sovellusohjelmalla varustettuina, jolloin lähetin-vastaanotinkin on siis jo toimintakykyinen. Asiakkaalle jää vain verkkoasetusten tekeminen eli laitteiden välisten loogisten kytkentöjen muodostaminen ja mahdollinen tietoliikenneparametrien hienosäätö. Tämä työvaihe tehdään tyypillisesti laitetta käyttökohteeseen asennettaessa.

Piirivalmistajan tehtaalta lähtevät Neuron-piirit sisältävät oletusasetukset 1,25 Mb/s lähetin-vastaanottimelle kierretyn parikaapelin verkkoon. Mikäli Neuron-piiri asennetaan laitteeseen, joka käyttää oletusasetuksista poikkeavaa lähetin-vastaanotinta, täytyy piiri ohjelmoida erillisellä ohjelmointilaitteella. Tämän avulla laitevalmistaja siirtää Neuron-piirille sovellusohjelman, joka sisältää tiedot laitteen käyttämästä lähetin-vastaanottimesta. Nämä asetukset mahdollistavat yhteyden Neuron-piiriin lähetin-vastaanottimen sisältämän kommunikaatioportin kautta, jolloin laite on hallittavissa verkonhallintatyökalulla.

Kun Neuron-piiri juotetaan laitteen piirilevyyn, ei ohjelmointilaitetta enää voida käyttää. Tällöin sovellusohjelman ja asetusparametrien muuttaminen onnistuu Neuron 3120 -piirissä ainoastaan Lon-verkkoa pitkin. Tämän yhteyden toimiminen puolestaan edellyttää, että asetukset ovat kohdallaan. Yhteys Neuron-piiriin voidaan menettää niin, ettei siihen enää saada yhteyttä verkon välityksellä. Tällöin laite muuttuu käyttökelvottomaksi eli käytännössä tuhoutuu. Neuron-piirin ja lähetin-vastaanottimen välisen yhteistoiminnan turvaamiseksi voidaan niiden parametreja asettaa Neuron 3120 -piirissä ainoastaan erillisellä ohjelmointilaitteella (Echelon 1995a). Tällä estetään valmiin laitteen tuhoutuminen verkon kautta ladattavan sovellusohjelman päivityksen sisältämien virheellisten asetusten takia.

Ulkoista muistia käyttävän Neuron 3150 -piirin sovellusohjelma voi sijaita ulkoisessa ROM- tai flash-muistissa. Myös tällöin sovellusohjelma sisältää lähetin-vastaanottimen tarvitsemat parametrit, mutta varusohjelma kopioi ne EEPROM-muistin kokoonpanon parametreihin. Kopiointi tapahtuu, kun integroidun ja ulkoisen muistin alkulataustunnukset eroavat toisistaan. EEPROM-muistin sisältämiä parametreja voidaan muuttaa verkon välityksellä, ja ne korvataan jälleen ulkoisen muistin parametreilla, mikäli ulkoisen muistin sisältö ja samalla alkulataustunnus muuttuvat. Valmiin laitteen virheellisten parametrien vuoksi menettämä verkkoyhteys voidaan näin ollen palauttaa, mikäli ulkoisen muistin sisältöä voidaan muuttaa ilman verkkoyhteyttä.

Ohjelmointityökalussa sovellusohjelmaan assosioidaan laitteiston malli, joka sisältää tiedot Neuron-piirin tyypistä ja kellotaajuudesta, lähetin-vastaanottimen tyypistä ja bittinopeudesta sekä varusohjelmiston versiosta. Lähetin-vastaanottimen kellosignaali muodostetaan Neuron-piirin sisään tulevasta kellosignaalista, joten molempien taajuus on merkittävä oikein.

Sovellusohjelmassa kokoonpanon parametreja voidaan käsitellä tietorakenteessa ”config_data_struct”, joka on määritelty Neuron C:n header-tiedostossa ”access.h”. Muuttuja ”config_data” on automaattisesti määritelty jokaisessa ohjelmassa, ja se sisältää käytössä olevat asetustiedot. Ne voidaan kopioida itse määriteltyyn muuttujaan, jota voidaan muokata ja jonka arvot voidaan päivittää käyttöön funktiolla ”update_config_data( )” (Echelon 2003a).

5.2.2 Sovelluskohtaiset parametrit

Käännetty sovellusohjelma sisältää myös sovelluskohtaisia parametreja. Niillä määritellään kiinteä verkkomuuttujien muistiosoitetaulukko, Neuron-piirin ja sovellusohjelman tunnisteet, mahdollisen standardiverkkomuuttujiin pohjautuvan laiterajapinnan dokumentaatio, verkkoalue-, osoite- ja verkkomuuttujataulukoiden merkintöjen lukumäärät sekä tietoliikenteen puskurimuistien koot ja lukumäärät. Piirin tunnusta ja tyyppiä sekä muistiosoitteita lukuun ottamatta nämä parametrit voidaan määritellä ohjelmakoodissa kääntäjän käskyillä. Nämä käskyt esitellään tässä dokumentissa kutakin taulukkoa tai puskuria esittelevässä kappaleessa. Sovellusohjelmassa määrittelemättä jätetyt parametrit saavat oletusarvon, jonka kääntäjä asettaa sisältämiensä laskukaavojen perusteella (Echelon 2003b).

Piirin tunnistetietoja lukuun ottamatta nämä parametrit kirjoitetaan uudelleen sovellusohjelmaa piirille ladattaessa. Parametrien asettamiseen ei ole tarjolla omia varusohjelmiston funktioita tai verkonhallintaviestejä. Verkonhallintakäskyillä pystytään kuitenkin kirjoittamaan sovellusohjelman käyttämälle muistialueelle, ja hallitusti tämä voidaan tehdä lataamalla uusi sovellusohjelma.

Sovellusohjelmassa nämä parametrit ovat luettavissa muuttujista ”read_only_data” sekä ”read_only_data2”, jotka toteuttavat rakenteet ”read_only_data_struct” ja ”read_only_data_struct_2” (Toshiba 2006). Rakenteet on määritelty Neuron C:n header-tiedostossa ”access.h”.

Sovelluskohtaiset parametrit määrittävät muistiosoitteiden jakamista eri objektien käyttöön. Tietoliikenteen asetustaulukoiden merkintäpaikkojen lukumäärät määräävät täsmällisesti asetustaulukoiden muistiosoitteet EEPROM-muistissa, kun taulukot sijaitsevat aivan EEPROM-muistin alussa ja merkintöjen koko on vakio (Toshiba 2006). Kiinteä verkkomuuttujataulukko sekä puskurimuistien määrittely sisältävät puolestaan tietoja piirin resetoinnin jälkeiseen RAM-muistin varaukseen.

5.2.3 RAM-muistin varaus

Resetoinnin yhteydessä Neuron-piirin varusohjelmisto varaa RAM-muistia järjestelmän käyttöön. Järjestelmämuisti varataan RAM-muistialueen alusta, ja se sisältää käyttöjärjestelmän vaatimien rakenteiden lisäksi tietorakenteet verkkoliikenteen ja I/O-tapahtumien puskurimuisteille sekä sovellusohjelman käyttämille ajastimille.

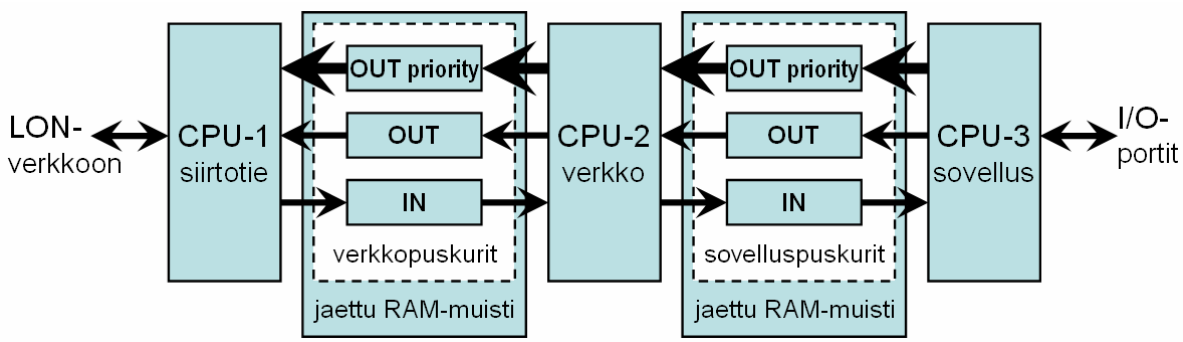
Sovellusohjelman käyttämä muisti varataan RAM-muistialueen lopusta alkua kohden. Muistialueiden riittäminen sovellusohjelman käyttöön on ohjelmointityökalun linkittäjän vastuulla, eikä varusohjelmisto tiedosta muistinkäytön kokonaistilannetta.

Verkonhallintatyökalulla voidaan muuttaa järjestelmämuistin kokoonpanoa, esim. jos halutaan muuttaa puskurimuistien kokoja. Verkonhallintaohjelma kykenee lukemaan järjestelmämuistille ennestään varatun RAM-muistin määrän, muttei tiedä vapaaksi jäävän muistin määrää ilman solmun .xif-tiedostoa. Se sisältää solmun ulkoisen verkkoliikenteen rajapinnan määrittelyjen lisäksi tiedon järjestelmämuistin suurimmasta sallitusta koosta. Tämän tiedon turvin järjestelmämuistin kokoa voidaan turvallisesti kasvattaa sovellusohjelman alkuperäisen linkityksen jälkeen. Ilman tätä tietoa järjestelmämuistin muutokset tulee tehdä niin, ettei sen tilantarve kokonaisuudessaan kasva.

5.3 Verkkoliikenteen puskurimuistit

Neuron-piiri sisältää kahdeksanbittiset sovellus-, verkko- ja siirtotieprosessorit, jotka yhdessä toteuttavat LonTalk-protokollan tiedonsiirtopalvelut. Näiden välinen yhteistoiminta tapahtuu puskurimuistien välityksellä. Keskimmäisenä sijaitseva verkkoprosessori kommunikoi alemman tason siirtotieprosessorin kanssa verkkopuskureiden välityksellä, ja ylemmän tason sovellusprosessorin kanssa sovelluspuskureiden välityksellä, kuva 4.

Neuron-piirille käännetty sovellusohjelma sisältää sovelluskohtaisia parametreja, joilla määritellään verkkoasetustaulukoiden kapasiteetin lisäksi puskurimuistien koot ja merkintäpaikkojen lukumäärät. Nämä asetetaan laitteen kehitysvaiheessa Neuron C -ohjelmakoodiin kääntäjän käskyillä. Puskurimuistien asetuksilla haetaan tasapainoa verkon suorituskyvyn ja muistialueiden säästymisen välillä. Puskurimuistit muodostavat osan järjestelmämuistista, joka kilpailee sovellusohjelman kanssa jaetun RAM-muistin alueista.



Kuva 4. Neuron-piirin prosessorit ja puskurimuistit.

Oletuksena Neuron-piirissä on puskurimuistipaikat kahdelle prioriteettitason ulostulolle, kahdelle normaalille ulostulolle sekä kahdelle sisääntulolle. Nämä muistipaikat löytyvät sekä sovellus- että verkkopuskureista. Verkkopuskurien muistialueet ainoastaan määritellään suuremmiksi, koska niiden kautta kulkevat viestipaketit ovat lähempänä fyysistä siirtotietä, jolloin pakettien siirtokehykset sisältävät enemmän LonTalk-protokollan vaatimia merkintöjä.

5.3.1 Tapahtumatiedon säilytys sovelluspuskureissa

Neuron-piirissä sovellusohjelman tehtävät ajoitetaan tapahtumien perusteella, jolloin when-lauseessa määritelty tapahtuma kontrolloi ohjelmakoodin ajoa. Neuron C:ssä määritellyt verkkoliikenteen tapahtumat sisältävät tiedon viestin vastaanottamisesta tai viestinlähetyksen onnistumisesta.

Neuron-piirissä sovellusohjelmaa ajava sovellusprosessori toimii asynkronisesti sille verkkoliikenteen palveluja tarjoavaan alempaan kerrokseen nähden. Mikäli jokin verkkoliikenteen tapahtuma liipaisee sovellusohjelman tehtävän, tulee tätä tapahtumatietoa säilyttää sovellusohjelmaa varten. Tarvittaessa sovellusohjelman tarvitsemaa tapahtumatietoa säilytetään sovelluspuskureissa.

Normaalisti käyttöjärjestelmän vuorotin vapauttaa automaattisesti sisääntulon sovelluspuskurin ja verkkomuuttujan arvo päivittyy ilman sovellusohjelman väliintuloa. Myös ulostulon sovelluspuskurit vapautuvat heti lähtevän viestin siirryttyä verkkoprosessorille, sillä verkkoprosessori tarjoaa luotettavan kuljetusyhteyden mahdollisine kuittauspalveluineen ilman sovellustason osallistumistarvetta. Mutta mikäli Neuron C -kääntäjä on havainnut sovellusohjelman käyttävän verkkoliikenteen tapahtumatietoa, säilytetään sovelluspuskurin sisältö tapahtumamerkintänä sovellusohjelmalle. Tällöin puskuri vapautuu vasta sovellusohjelman käsiteltyä tapahtuman.

Fyysisesti verkkoliikenteen tapahtumatiedot sijaitsevat siis sisään- ja ulostulon sovelluspuskureissa. Ne tarjoavat sovellusohjelmalle tiedon viestin vastaanottamisesta tai viestinlähetyksen onnistumisesta. Loogisesti nämä tapahtumatiedot muodostavat kaksi jonoa, joiden sisällöt on pakko käsitellä järjestyksessä. Mikäli sovellusohjelma ei käsittele jonon kärjessä olevaa tapahtumatietoa, jonon käsittely pysähtyy eikä vastaavia sovelluspuskureita voida vapauttaa. Tällöin tapahtumien normaali käsittely estyy eikä laite kykene reagoimaan myöhempiin sovellustason tai verkonhallinnan viesteihin. Ongelma voi esiintyä, mikäli tehtävien ajoituksen kiertovuorottelun nollaus on käytössä. Tällöin vuorotin aloittaa aina ensimmäisenä esiintyvästä when-lauseesta verkkomuuttujan päivittyessä tai ajastimen rauetessa. Vuorottimen liian tiheä nollaus voi aiheuttaa vapaiden sovelluspuskureiden loppumisen, jos ohjelmakoodissa myöhemmin esiintyvää when-

lausetta ei ehditä tarkastaa. Jos sovellusohjelma käyttää jotakin tapahtumatietoa, on mahdollinen tapahtuma siis tarkistettava säännöllisesti.

Sovellusviestejä käytettäessä tulee sovellusohjelman aina huomioida saapuneet viestit säännöllisesti sisääntulopuskureiden vapauttamiseksi, koska ne eivät vapaudu ilman sovellusohjelman huomiota automaattisesti kuten verkkomuuttujia käytettäessä.

5.3.2 Puskureiden merkintäpaikkojen tarve

Sisääntulopuskureiden lukumäärän tarve riippuu käytetystä tiedonsiirtopalvelusta sekä solmujen välisistä loogisista kytkennöistä. Lukumäärien oletusarvot riittävät yksittäisten solmujen väliseen viestintään, mutta eivät ryhmän jäsenten välisiin kytkentöihin käytettäessä kuittaavaa tai kyselypalvelua. Tällöin sisääntulon verkkopuskurissa tulisi olla merkintäpaikat kaikille ryhmän jäsenten lähettämille vastauksille, jotka saapuvat lähes samanaikaisesti. Suurissa kytkennöissä toistava tiedonsiirtopalvelu onkin kuittaavaa parempi, koska sillä saavutetaan sama luotettavuus vähemmällä verkkoliikenteellä, eikä se aseta vaatimuksia sisääntulopuskureille. Sisääntulon verkkopuskurien tarve riippuu toisaalta myös verkon bittinopeudesta ja Neuron-piirin kellotaajuudesta, ja puskurien minimimäärän löytäminen saattaa edellyttää käyttökokeita asennetussa verkossa.

Sovellusviestien kysely/vastauspalvelua käytettäessä myös sovelluspuskurin tulee sisältää sisääntulomuistipaikkoja jokaiselle samaan pyyntöön vastauksena saapuvalle viestille, tai muutoin osa niistä saattaa jäädä huomiotta. Varatut sovelluspuskurit vapautuvat vasta kun sovellusohjelma on käsitellyt saapuneiden vastausviestien generoimat tapahtumat.

Ulosmenevän liikenteen sovelluspuskureiden lukumäärää on syytä kasvattaa käytettäessä synkronista tiedonsiirtoa, koska tällöin ulostuloverkkomuuttuja saattaa saada päivityksiä nopeammin kuin niitä ehditään lähettää, ja tässä palvelumuodossa kaikki päivitykset lähetetään. Näin vältetään sovellusohjelman ajautuminen estotilaan, jolloin se suorittaa vain tapahtumat liittyen viestien vastaanottamiseen ja viestinlähetyksen onnistumisen kontrollointiin. Ohjelman ajo rajoittuu näiden tapahtumien käsittelyyn, kunnes ulostulon sovelluspuskureita vapautuu, mikä aiheuttaa viiveen muiden tehtävien suorittamiseen. Estotilan kesto riippuu verkon kuormituksesta ja tiedonsiirtokanavan bittinopeudesta.

Mikäli Neuron C -kääntäjä ei löydä ohjelmakoodista verkko- ja sovelluspuskurit asettavia kääntäjän käskyjä, muodostaa se puskurimuistit oletusarvojen mukaisesti. Seuraavana on esitetty sovellus- ja verkkopuskureiden merkintäpaikkojen lukumäärien asettamiseen käytettävät kääntäjän käskyt oletusarvoilla varustettuina.

```
#pragma app_buf_in_count                2
#pragma app_buf_out_count               2
#pragma app_buf_out_priority_count      2
#pragma net_buf_in_count                2
#pragma net_buf_out_count               2
#pragma net_buf_out_priority_count      2
```

5.3.3 Vastaanottotapahtumien puskurimuisti

LonTalk-protokollan kuljetuskerroksen palveluihin kuuluu viestien kaksoiskappaleiden tunnistaminen. Viestin kahdentuminen saattaa aiheutua verkon heijastuksista tai toistavan kuittauksettoman tiedonsiirtopalvelun toistoista. Kaksoiskappaleiden suodattamiseksi vastaanotettujen viestien tunnistetietoja säilytetään omassa puskurimuistissaan vastaanottoajastimen osoittama aika. Näin määrätyn ajan sisällä saapuvat identtiset viestit voidaan tunnistaa.

Verkkoprosessori vertaa saapuvan viestin tunnistetietoja puskurimuistin merkintöihin, ja tunnistetietojen vastatessa toisiaan saapunut viesti jätetään huomiotta. Jos taas saapuneen viestin tunnistetietoja ei vielä löydy vastaanottotapahtumien puskurimuistista, tulkitaan viesti ensimmäistä kertaa vastaanotetuksi ja sen tunnistetiedot lisätään puskuriin. Tapahtumamerkintää käytetään kaikissa tiedonsiirron palvelumuodoissa kertaviestejä lukuun ottamatta, ja uusi merkintä tehdään tunnistetietojen hiemankin poiketessa. Tunnistetietoina toimivat tapahtumanumero, lähde- ja kohdeosoite sekä prioriteetti.

Puskurimuistin kapasiteetti määrää, kuinka montaa eri vastaanottotapahtumaa solmu voi käsitellä samanaikaisesti. Jokainen puskurin merkintä vaatii 13 tavua tilaa RAM-muistista, ja merkintöjen lukumäärä voidaan asettaa yhdestä 16:een seuraavalla kääntäjän käskyllä:

```
#pragma receive_trans_count          n
```

Mikäli Neuron C -kääntäjä ei löydä käskyä ohjelmakoodista, käyttää se puskurimuistin kokona oletusarvoa. Oletusarvolle on määritelty laskukaava (Echelon 2003b), joka huomioi verkkomuuttujien lukumäärän sekä mahdollisen sovellusviestien käytön.

5.3.4 Puskureiden kokojen määritys

Verkkoliikenteen puskurimuistien oletuskoot riittävät verkkomuuttujia käytettäessä. Tällöin Neuron C -kääntäjä valitsee puskureiden koot suurimman määritellyn verkkomuuttujan koon perusteella. Sovellusviestejä käytettäessä sovellusohjelman kirjoittajan on varmistuttava, että puskurit pystyvät käsittelemään suurimmatkin sovellusohjelman ja varusohjelmiston generoimat ja vastaanottamat viestit. Tämä saattaa vaatia puskurimuistien koon kasvattamista Neuron C -ohjelmakoodissa kääntäjän käskyillä.

Sekä verkkomuuttujien päivitysviestit että sovellusviestit voidaan varustaa eksplisiittisellä viestin kohdeosoitteella. Tällöin osoitteen sisältävän tietorakenteen kentät täytetään sovellusohjelmassa viestin lähetyksen yhteydessä. Osoitteen käsittelyn siirtyessä näin verkkokerrokselta sovellusohjelmalle kasvavat myös sovelluspuskurin kokovaatimukset. Eksplisiittistä osoitetta käytettäessä sovelluspuskuri sisältää 11-tavuisen osoitekentän (Toshiba 2006), ja sovelluspuskureille suositellaan samaa kokoa kuin verkkopuskureille.

Puskurimuistien kokojen oletusarvojen laskukaavat sekä muut sallitut arvot on listattu ohjelmointioppaassa (Echelon 2003b). Oletusarvojen laskennassa huomioidaan verkkoliikenteessä käytettävä viestintämuoto sekä osoitteenmuodostusperiaate. Verkkopuskurien minimikoot määräytyvät mahdollisesti vastaanotettavista ja generoituvista verkonhallintaviesteistä, ja Neuron C -kääntäjä varoittaa näitä minimiarvoja rikkovista kääntäjän käskyistä. Seuraavana on esitetty verkko- ja sovelluspuskureiden kokojen asettamiseen käytettävät kääntäjän käskyt pienimmillä suositeltavilla arvoilla varustettuina.

```
#pragma app_buf_in_size      22
#pragma app_buf_out_size     20
#pragma net_buf_in_size      50
#pragma net_buf_out_size     42
```

Määrittelemällä tiedonsiirron puskurimuistien koot ja lukumäärät sopiviksi voidaan siis parantaa solmun ja verkon suorituskykyä ja järkeistää solmun rajallisen muistikapasiteetin käyttöä. Sovellusohjelman kirjoittajan tulee myös huolehtia, että kaikki ohjelmakoodissa esitellyt tapahtumat tarkastetaan säännöllisesti sovelluspuskureiden vapauttamiseksi.

6 VERKKOLIIKENTEEN ASETUSTAULUKOT

Jokaisen Neuron-piirin EEPROM-muistissa on verkkoliikenteen asetustaulukot solmun omalle osoitteelle, viestien kohteina olevien solmujen osoitteille sekä verkkomuuttujille.

Verkkoaluetaulukko sisältää solmun oman loogisen osoitteen suhteessa muuhun Lon-verkkoon. Osoitetaulukko sisältää luettelon kohdeosoitteista, joihin solmu lähettää viestejä. Verkkomuuttujien tiedot ovat jakautuneet muistiosoitteet sisältävään kiinteään taulukkoon, solmujen väliset loogiset kytkennät määrittävään asetustaulukkoon sekä verkkomuuttujalle ylimääräiset tunnukset mahdollistavaan aliastaulukkoon.

Neuron-piirin verkkoliikenteen asetustaulukoiden tunteminen mahdollistaa valmistajakohtaiset ratkaisut, joissa verkkoasetuksia ei aseteta verkonhallintatyökalulla, vaan Neuron-piirin sovellusohjelma asettaa solmun verkkoasetukset automaattisesti. Tällöin verkon rakenne on määritelty jo sovellusohjelmaa kirjoitettaessa. Tämän diplomityön ohessa valmistunut hissinohjausjärjestelmän simulointiympäristö sisältää kuvaillun kaltaisen valmistajakohtaisen Lon-verkon.

Asetustaulukoiden rakenteiden ja käyttötarkoitusten tuntemista voidaan myös hyödyntää, kun tiedetään kuinka suuret verkkoalue- ja osoitetaulukot on varattava. Oletuksena niille varataan maksimikoot, mutta kääntäjän käskyillä voidaan varattujen muistipaikkojen määrää karsia. Näin voidaan tehostaa solmun rajallisen muistikapasiteetin käyttöä. Lisäksi asetustaulukot sisältävät LonTalk-protokollan parametreja, joiden tunteminen mahdollistaa verkon suorituskyvyn optimoinnin ja ongelmakohtien löytämisen.

6.1 Menetelmät verkkoasetusten tekemiseksi

LonWorks-laitteen asentaminen verkkoon sisältää laitteen fyysisen liittämisen tiedonsiirtokanavaan sekä verkkoliikenteen parametrien täyttämisen EEPROM-muistin asetustaulukoihin.

Verkkoasetukset voidaan asettaa joko valmiiksi tehtaalla, paikallisesti solmun sovellusohjelmassa tai verkon välityksellä verkonhallintaohjelmalla. Asetuksia tehdään ennen laitteen fyysistä kytkemistä, fyysisen kytkemisen yhteydessä sekä myöhemmin verkon kokoonpanossa tapahtuvien muutosten yhteydessä.

Verkkoasetusten tekemisessä käytettävä menetelmä ratkaisee verkon muodostamisen ja laitteen kytkemisen helppouden. Toisaalta se vaikuttaa mahdollisuuteen muunnella verkon rakennetta sekä kykyyn asentaa eri valmistajien tuotteita samaan verkkoon.

6.1.1 Esiasennettu verkko

Esiasennetussa verkossa kaikki verkkoasetukset asetetaan laitteisiin jo tehtaalla. Asetukset määritetään jo tuotteen suunnittelun yhteydessä, ja tehtaan ohjelmointilaitteella asetukset kopioidaan kaikkiin tuotantolinjalta valmistuviin identtisiin laitteisiin.

Esiasennetun verkon laitteiden asentaminen on suoraviivaista käyttökohteessa, mutta verkon rakenne on tiedettävä tarkasti ennalta. Verkon solmujen lukumäärä tai tehtävät eivät saa muuttua tuotteen elinkaaren aikana. Tämä onnistuu kenttäväylän sisältävässä tuotteessa, esim. painokoneessa tai autossa.

6.1.2 Itseasentuva laite

Itseasentuvassa verkossa jokainen verkon solmu päivittää itse omat verkkoasetuksensa, eikä erillistä verkonhallintaohjelmistoa tarvita. Itseasentuvan laitteen tärkeimmät osoitetiedot asetetaan siihen paikallisella syötteellä. Nämä asetukset voidaan tehdä vasta käyttökohteessa, mikä mahdollistaa variaatiot verkon rakenteeseen. Verkko-osoitteet voidaan valita käyttökohteen tarpeiden mukaisesti esim. vastaamaan fyysistä merkitystä kuten kerroksien tai huoneiden numeroita. Verkon kokoonpanon muuttuessa voidaan osoitteita myös asettaa uudelleen, mikä mahdollistaa verkon laajennukset tai verkkojen yhteenliittämisen.

Itseasentuvassa laitteessa Neuron-piirin sovellusohjelma käyttää varusohjelmiston funktioita verkkoliikenteen asetustaulukoiden kirjoittamiseen. Sovellusohjelma valitsee oikeat asetukset vastaanottamiensa syötteiden perusteella. Osoitetietoja voidaan syöttää solmun sovellusohjelmalle Neuron-piirin I/O-rajapinnan välityksellä esim. DIP-kytkimellä, sarjaliitännällä, infrapunayhteydellä. Erikoislaite mahdollistaa tietojen syötön myös Lon-verkon rajapinnan kautta (Motorola 1997a).

Sovellusohjelma vastaanottaa osoitetiedot tyypillisesti Neuron-piirin I/O-rajapinnan kautta, jonka resursseja jää näin vähemmän varsinaisen sovelluksen käyttöön. Fyysiset kytkimet tai liitännät nostavat myös laitteen valmistuskustannuksia. Kaikki osoitetietoja vastaanottavat mekanismit kasvattavat sovellusohjelman kokoa, jonka lisääntynyt monimutkaisuus voi huonontaa Neuron-piirin I/O-toimintojen vasteaikoja. Lisäksi menetelmät ovat usein laitevalmistajien tuotekohtaisesti räätälöimiä, mikä vaikeuttaa eri valmistajien laitteiden ja verkkojen välistä yhteentoimivuutta.

Itseasentuvat laitteet sopivatkin lähinnä pieneen verkkoon, jonka solmut voidaan vielä tavoittaa fyysisesti asetusten tekemiseksi. Yhdessä kanavassa toimiva verkko ei tarvitse reititintä, jolloin laitteet voivat käyttää samoja ajastimien ja viiveiden arvoja tiedonsiirrossa. Tällöin I/O-rajapinnan kautta syötettävien parametrien lukumäärä voidaan pitää kohtuullisena.

6.1.3 Verkon ylläpito verkonhallintaviesteillä

Suurin mahdollinen joustavuus Lon-verkon muodostamisessa ja ylläpidossa saavutetaan verkonhallintaohjelmaa käyttämällä. Verkonhallintaohjelma käyttää LonTalk-protokollassa määriteltyjä verkonhallintaviestejä asetusten tekemiseksi muihin verkon solmuihin. Tällöin verkkoon kytkettyä laitetta ei tarvitse tavoittaa fyysisesti sen asetusten muuttamiseksi, vaan kaikki asetukset voidaan tehdä verkon välityksellä.

Verkonhallinnassa yksi laite tai sovellus ylläpitää tietokantaa verkon asetuksista ja pyrkii muuttamaan verkon sisältämien laitteiden asetuksia yhdenmukaisiksi tietokannan kanssa (Echelon 2006). LonWorks-tekniikan alkuaikoina oli tarjolla lisäprosessorilla varustettuja verkonhallintalaitteita, jotka operoivat verkonhallintaviesteillä ja ylläpitivät verkon asetusten tietokantaa (Motorola 1997a). Nykyään verkonhallinta suoritetaan pääsääntöisesti PC:n ohjelmistolla, joka kytkeytyy Lon-verkkoon verkkosovittimen välityksellä. Verkonhallintaohjelmalla verkko muodostetaan helposti yhdistämällä solmut graafisen käyttöliittymän kautta. Verkonhallintaohjelma muodostaa tällöin tietokannan, jonka avulla verkon asetuksia voidaan kätevästi myöhemminkin muuttaa.

Lon-verkko muodostaa hajautetun järjestelmän, jonka jäsenet kykenevät itsenäiseen toimintaan. Jokainen Neuron-piiri kykenee myös verkonhallintaviestien lähettämiseen, ja näin niillä on valmius verkonhallintatoimintojen suorittamiseen. Verkonhallintatoiminnot on kuitenkin perusteltua keskittää yhteen yksikköön tai

ainakin käyttää keskitettyä tietokantaa verkon asetuksille. Verkon tietokannan ylläpito vaatii runsaasti muistia ja laskentatehoa. Muista verkon solmuista saadaan yksinkertaisempia, tehokkaampia ja halvempia, kun ne vapautetaan verkkoasetusten hallintatehtävistä. Keskitettyä verkkoasetusten tietokantaa käytettäessä ajantasaiset tiedot löytyvät aina kootusti yhdestä paikasta.

Lon-verkon laitteiden tyypilliset ohjaus- tai valvontatehtävät eivät tarvitse verkonhallintaohjelman tai verkkoasetusten tietokannan palveluja. Niitä tarvitaan verkon muodostamiseen sekä lisättäessä solmuja verkkoon tai muutettaessa niiden välisiä loogisia kytkentöjä. Verkonhallintaohjelmaa käytetään lisäksi verkon toiminnan tarkkailuun ja vianpaikannukseen. Verkon suorituskykyä voidaan optimoida asettamalla tiedonsiirtoa ohjaavat ajastimet vastaamaan verkon fyysisiä viiveitä, jotka vaihtelevat eri puolilla verkkoa kytkennän tyypistä ja solmujen fyysisestä sijainnista riippuen.

Verkonhallintaviestejä käyttämällä voidaan asettaa lähes kaikki Neuron-piirin parametrit. Verkkoliikenteen asetustaulukoiden merkinnät voidaan kirjoittaa helposti omilla verkonhallintaviesteillään. Lisäksi käytössä on kohdepiirin muistin lukemisen ja kirjoittamisen mahdollistavat viestit. Näiden avulla kohdepiiriä voi periaatteessa hallita hyvin monipuolisesti, mutta käytännön esteenä on kohdepiirin sisäisten riippuvuussuhteiden hallitsemisen monimutkaisuus. Esimerkiksi Neuron-piirin muistialueiden jaon määrittävät sovelluskohtaiset parametrit asetetaan siirtämällä kohdepiiriin uudelleen käännetty versio sovellusohjelmasta, sillä vain ohjelmointityökalun linkittäjä hallitsee kohdepiirin fyysisten muistialueiden käytön kokonaistilanteen.

Verkonhallintaviestien yhteydessä voidaan käyttää LonTalk-protokollan aidonnuspalvelua, jolloin vastaanottava solmu reagoi niihin vasta aidonnuksen onnistuttua. Näin estetään luvaton solmujen manipulointi. Kokoonpanon parametri ”nm_auth” määrittelee, käytetäänkö aidonnusta verkonhallintaviestien yhteydessä. Asentamaton solmu ei käytä aidonnustoimintoa, ja se kytkeytyy päälle vasta asetettaessa laite normaaliin tai yhteydettömään toimintatilaan. Mikäli aidonnusta on käytetty aiemmin, on tällöin varottava aidonnuksen tahatonta päälle kytkeytymistä. Sen parametri on saattanut jäädä asetetuksi, mikä huomataan vasta aidonnuksen kytkeytyessä päälle piirin toimintatilan muutoksen yhteydessä.

Sovellusohjelma sisältää parametrin ”read_write_protect”. Mikäli tämän asetuksen sisältävä ohjelmakoodi siirretään Neuron-piiriin, reagoi se enää rajoitetusti verkonhallinnan luku- ja kirjoitusviesteihin. Tämän jälkeen piiristä voidaan lukea standardien verkkomuuttujien, kokoonpanon ja sovelluksen parametrit sekä muut sovellusohjelman muistialueet. Ainoastaan lähetin-vastaanottimen toimintaa määrittelevien kokoonpanon parametrien kirjoittaminen on mahdollista. Tätä asetusta ei voida perua verkon välityksellä, joten toiminto soveltuu vain valmiiden tuotteiden suojaamiseen.

LonWorks-tekniikan tarjoamat ominaisuudet tulevat täysin hyödynnettyä vasta tehtäessä verkkoasetukset verkonhallintaviesteillä. Verkonhallintaohjelma tarjoaa varsinkin suurempiin verkkoihin parhaan joustavuuden asetusten ja loogisten kytkentöjen tekemiseksi. Vaikka menetelmästä tulee kustannuksia verkonhallintaohjelmiston lisenssin kautta, syntyy säästöjä helpomman ylläpidon ja verkkoasetusten hallintatehtävistä vapautettujen yksinkertaisempien solmujen myötä. Verkonhallintatyökaluilla tapahtuva verkon asennus ja ylläpito edustavatkin LonWorks-tekniikan tyypillisintä käyttötapaa mm. rakennusautomaatiossa.

6.1.4 Verkkoasetusten teko varusohjelmiston funktioilla

Verkkoliikenteen asetustaulukoiden merkintäpaikkojen lukumäärät määritellään sovellusohjelmassa, ja nämä arvot ovat luettavissa myös verkonhallintaohjelmalla. Yhdistämällä tieto varatuista resursseista käytetyn piirin ja varusohjelmiston tyyppitietoihin voidaan asetustaulukoiden muistiosoitteet selvittää. Neuron-piirin integroituun EEPROM-muistiin on mahdollista kirjoittaa verkonhallintaohjelmalla, ja asetusten suora binääritason manipulointi on siis mahdollista. Tätä ei kuitenkaan turvallisuussyistä suositella. Sen sijaan Neuron-piirin varusohjelmisto tarjoaa joukon valmiita funktioita, joilla verkkoasetukset voidaan tehdä hallitusti. Sovellusohjelmassa verkkoasetusten teon mahdollistavat funktiot muistuttavat vastaavia verkonhallintaviestejä ja sisältävät solmun jumiutumisen estäviä suojamekanismeja.

Itseasentuvassa laitteessa Neuron-piirin sovellusohjelma käyttää varusohjelmiston funktioita verkkoliikenteen asetustaulukoiden kirjoittamiseen. Myös esiasennettuja laitteita voidaan toteuttaa samalla menetelmällä, mikäli käytettävissä ei ole Neuron-piirin ohjelmointilaitetta eikä riittävän monipuolista verkonhallintaohjelmaa, joilla asetukset voitaisiin kirjoittaa suoraan EEPROM-muistin taulukoihin. Tällöin verkkoasetukset voidaan tehdä turvallisesti sovellusohjelmassa varusohjelmiston funktioilla. Esiasennetun ja itseasentuvan laitteen eroksi jää tällöin, että ainoastaan itseasentuva laite huomioi tekemissään asetuksissa vastaanottamansa paikalliset syötteet.

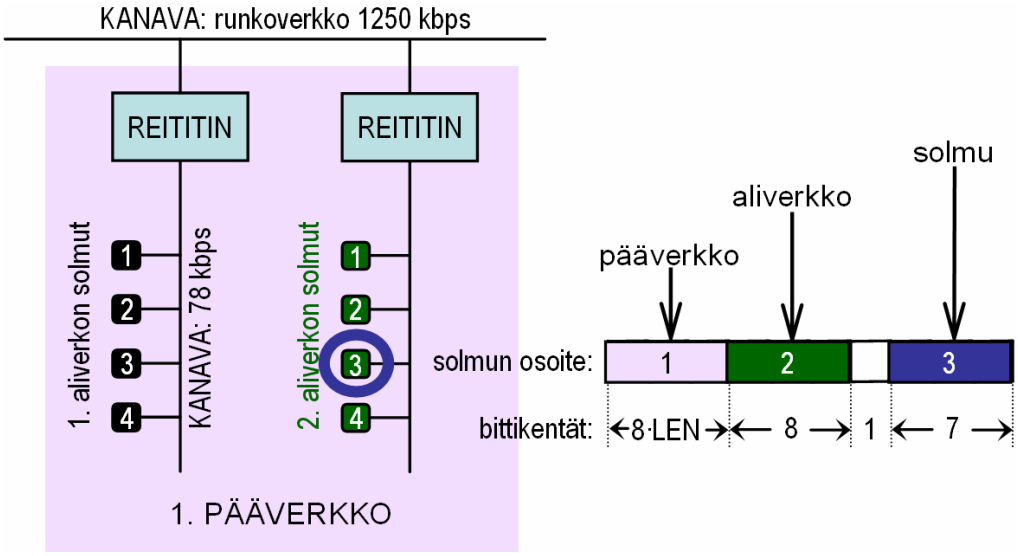
Uudet verkkoasetukset käyttöön ottavat funktiot sisältävät varmistusmekanismin, joka estää resetoinnin tai jännitehäiriön aiheuttaman asetusten tuhoutumisen EEPROM-muistin kirjoituksen aikana. Mekanismi ylläpitää virhelokitietoja EEPROM-muistissa, joita kirjataan aina kutsuttaessa funktioita ”update_program_id( )”, ”update_config_data( )”, ”update_domain( )”, ”update_clone_domain( )”, ”update_address( )”, ”update_nv( )” tai ”update_alias( )”. Lokitiedon kirjaus tapahtuu aina näitä funktioita kutsuttaessa, vaikka varsinaista asetuksen kirjoitusta ei tapahtuisikaan lähteen ja kohteen identtisyyden takia. Näiden funktioiden tarpeetonta kutsumista tulee välttää sovellusohjelmassa, jotta EEPROM-muistin rajallinen käyttöikä riittäisi.

6.2 Verkkoaluetaulukko

Tieto solmun omasta loogisesta osoitteesta suhteessa muuhun Lon-verkkoon sijaitsee verkkoaluetaulukossa. Se sisältää tiedot solmun pääverkosta, aliverkosta ja solmun tunnuksesta aliverkon sisällä, kuten kuva 5. esittää.

Solmu voi kuulua samanaikaisesti kahteen pääverkkoon, jolloin verkkoaluetaulukkoon tulee merkinnät myös toisen verkkoalueen asetuksille. Kumpikin taulukon merkintä vaatii 15 tavun tilan EEPROM-muistista. Laitteen hyväksyminen LonMark-yhteensopivaksi edellyttää kahden verkkoaluetaulukon paikan varaamista, ja oletuksena ne ovatkin varattuina. Tarpeettoman taulukon viemä muistitila voidaan vapauttaa seuraavalla kääntäjän käskyllä:

```
#pragma num_domain_entries        1
```



Kuva 5. Solmun looginen osoite ja bittikenttien pituudet.

Verkkoaluetaulukon yhden merkinnän sisältämät tiedot kuvataan rakenteessa ”domain_struct”. Määrittely löytyy Neuron C -kielen header-tiedostosta ”access.h”. Echelonin Nodebuilder-ohjelmistossa se sijaitsee oletuksena hakemistossa ”C:\LonWorks\NeuronC\Include\”.

```
typedef struct domain_struct {
        unsigned        id [DOMAIN_ID_LEN]; // DOMAIN_ID_LEN = 6
        unsigned        subnet;
        unsigned        not_clone       : 1; // 0 => clone domain; read-only
        unsigned        node            : 7;
        unsigned        len;                 // May only have values 0, 1, 3 or 6
        unsigned        key [AUTH_KEY_LEN]; // AUTH_KEY_LEN = 6
} domain_struct;
```

EEPROM-muistin taulukossa pääverkon tunnukselle (Domain ID) varataan aina kuusi tavua tilaa, vaikka parametri ”len” määrittelee viesteihin kirjoitettavan tunnuksen pituuden. Mikäli pituuden arvona on 0, tulkitaan myös pääverkon tunnukseksi 0. Pituuden arvo 0xFF ilmaisee, ettei tunnusta ole lainkaan määritelty.

Aliverkon tunnukselle ”subnet” on varattu kahdeksan bittiä ja solmun omalle tunnukselle ”node” seitsemän bittiä. Aliverkon tunnuksessa 0 toimii yleislähetyksen symbolina, eli silloin vastaanottajina toimivat kaikki pääverkon solmut. Lon-verkon hierarkiaa esitellään luvussa 3.3.

Verkkoaluetaulukon viimeinen kenttä sisältää kuuden tavun pituisen salausavaimen, jota käytetään LonTalk-protokollan aidonnuspalvelussa, ks. luku 3.5.

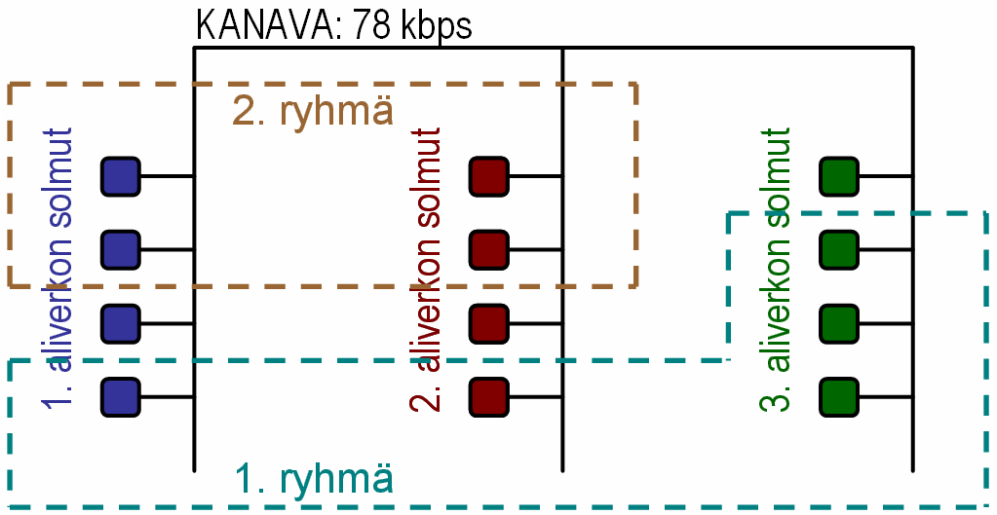
Sovellusohjelma voi kopioida verkkoaluetaulukon merkinnän sovelluksessa määriteltyyn domain_struct-tyypin muuttujaan funktiolla ”access_domain( )”. Tätä kopiota voidaan muokata ja uudet arvot voidaan päivittää käyttöön funktiolla ”update_domain( )” (Echelon 2003a).

Verkkoaluetaulukon merkinnässä bittikenttä “not_clone” ilmaisee, onko kyseessä kloonilaite. Tätä bittikenttää voidaan vain lukea, sillä sen arvo asetetaan funktioilla ”update_domain( )” ja ”update_clone_domain( )”. Käytössä oleva arvo voidaan lukea funktion ”access_domain( )” palauttamasta struct-rakenteesta.

6.2.1 Kloonilaite

Suoritettaessa verkkoaluetaulukon uusien tietojen päivitys funktiolla ”update_clone_domain( )” muodostuu kloonilaite. Se pystyy vastaanottamaan viestit, vaikka lähettäjän osoite on sama kuin vastaanottajan. Tällöin kloonilaitteille voidaan asettaa kiinteä solmun tunnus, jonka lukuarvo on esim. 1 kaikissa verkon kloonilaitteissa. Koska solmuilla ei ole yksilöllistä loogista osoitetta, ei niiden asentamiseen tarvita verkonhallintatyökalua. Näin voidaan luoda halpoja järjestelmiä.

Myös kloonilaitteen aliverkon tunnus voi olla kiinteä, tai sen arvo voidaan asettaa I/O-portin kautta tulevan paikallisen syötteen perusteella. Tähän voidaan käyttää mm. painonappeja, DIP-kytkimiä tai valintalevyä. Aliverkoista voidaan näin muodostaa ryhmiä, ja tätä ryhmittelyä voidaan käyttää varsinaisesta ryhmätunnuksesta riippumatta. Näin kloonilaitteista voidaan muodostaa ryhmiä sekä aliverkkotunnusten perusteella että tavanomaisemmin osoitetaulukon ryhmälähetysmäärittelyllä. Kuva 6. havainnollistaa jaottelua. Myös osoitetaulukkoon määritellyt ryhmät voidaan asettaa kloonilaitteisiin paikallisesti I/O-portin syötteellä. Reitittimiä ei suositella itseasentuviin laitteisiin pohjautuvaan verkkoon, mikä rajoittaa järjestelmän laajuuden yhteen kanavaan (Echelon 2006).



Kuva 6. Kloonilaitteet muodostavat ryhmiä aliverkon ja ryhmätunnuksen perusteella.

Aliverkon muodostaman ryhmän jäsenet tavoitetaan aliverkon sisäisellä yleislähetyksellä. Kaikki pääverkon solmut tavoitetaan pääverkon yleislähetyksellä. Osoitetaulukkoon määritellyn ryhmän jäsenet viestivät keskenään ryhmälähetyksellä. Aliverkon ja solmun tunnuksella ei sen sijaan voida osoittaa kloonilaitteita. Neuron-piirin ID-tunnukseen pohjautuva osoitemuoto toimii kloonilaitteissa, mutta vaikean ylläpidettävyyden takia sen käyttöä ei suositella. Osoitetyypit on esitelty tarkemmin seuraavassa luvussa.

Tiedonsiirtopalveluista vain toistetut ja kertaviestit ovat virallisesti tuettuja kloonilaiteissa. Kloonilaite ei kykene vastaanottamaan kuittauksia tai kysely/vastauspalvelun vastausviestejä. Se kykenee kuitenkin itse näiden lähettämiseen, mikäli se on varustettu riittävillä osoitetiedoilla (Echelon 2003c). Aidonnusta ei voida käyttää, koska tämän palvelun generoimat vastausviestit käyttävät aina aliverkon ja solmun loogista osoitetta.

Mikäli verkko sisältää identtisiä kloonilaitteita, saattaa lähettävä solmu tahattomasti päivittää myös omaa sisääntuloverkkomuuttujaansa. Näin voi käydä, koska lähettävä laite sisältää saman sisääntuloverkkomuuttujan kuin muut solmut, ja se saattaa

vastaanottaa heijastuksen toisille kloonilaitteille lähettämästään päivitysviestistä (Echelon 2003c). Ongelmaa ei esiinny heterogeenisissä kloonilaitteissa, joissa saman solmun sisään- ja ulostuloverkkomuuttujilla on eri valintatunnukset. Näiden kloonilaitteiden muodostama järjestelmä voidaan jakaa jokaisen verkkomuuttujan osalta lähettäviin ja vastaanottaviin solmuihin.

6.3 Osoitetaulukko

Osoitetaulukko sisältää luettelon kohdeosoitteista, joihin taulukon sisältävä solmu lähettää verkkomuuttujan päivitystietoja, päivityskyselyjä tai sovellusviestejä. Se myös määrittelee, mihin ryhmiin solmu kuuluu. Tyypillisesti verkonhallintaohjelma asettaa osoitetaulukon sisällön muodostettaessa verkon solmujen väliset loogiset kytkennät.

Jokainen Neuron-piiri sisältää oman valmistusvaiheessa annetun yksilöllisen 48-bittisen tunnuskoodin. Tätä tunnuskoodia käytetään verkko-osoitteena vain verkkoa muodostettaessa, sen asetuksia muutettaessa tai käytettäessä eksplisiittistä osoitemuotoa. Tavallisesti käytetään loogista osoitetta, joka muodostuu pääverkon ja aliverkon osoitteesta sekä solmun omasta tunnuksesta aliverkon sisällä. Mikäli solmu kuuluu useamman solmun välisen kytkennän muodostamaan ryhmään, korvataan osoitteessa aliverkon ja solmun tunnukset ryhmän tunnuksella. Viestejä voidaan lähettää myös kaikille pääverkon solmuille tai tietyn aliverkon solmuille. Taulukko 3. esittää viestipakettien osoitteenmuodostuksen vaihtoehdot sekä niiden tallennukseen käytettävät tietorakenteet.

Taulukko 3. Osoitteenmuodostuksen vaihtoehdot ja osoitetaulukon tyyppi.

OSOITTEEN RAKENNE	OSOITETUT SOLMUT	TAULUKKO
pääverkko, aliverkko, ID-tunnus	yhden solmun fyysinen osoite	solmulähetys
pääverkko, aliverkko, solmu	yhden solmun looginen osoite	solmulähetys
pääverkko, ryhmä	kaikki ryhmän solmut	ryhmälähetys
pääverkko, aliverkko	kaikki aliverkon solmut	yleislähetys
pääverkko, aliverkko = 0	kaikki pääverkon solmut	yleislähetys
-	solmun sisäinen kytkentä	sisäinen kytkentä

Osoitetaulukossa tulee olla kohdeosoitteen merkintä jokaiselle ulostuloverkkomuuttujalle ja lähtevälle sovellusviestille. Sisääntulomuuttujaan assosioitua kohdeosoitetta käytetään kyselypalvelussa sekä ryhmälähetyksessä. Mikäli useampi verkkomuuttuja lähettää viestejä samaan kohteeseen, voivat ne käyttää yhteistä osoitetaulukon merkintää.

Osoitetaulukkoon varataan oletuksena tilaa 15:lle eli maksimimäärälle kohdeosoitteen merkintöjä. Jokainen varattu paikka vie viisi tavua tilaa EEPROM-muistista. Mikäli tarvittavien merkintöjen lukumäärää osataan arvioida, voidaan osoitetaulukon merkinnöille varatut paikat rajoittaa lukumäärään n seuraavalla kääntäjän käskyllä:

```
#pragma num_addr_table_entries      n
```

Osoitetaulukon bittikentän tulkita riippuu osoitettavien kohteiden lukumäärästä ja kokoonpanosta. Tietorakenteet on määritelty taulukon 3. mukaisesti erikseen yleislähetykselle, ryhmälähetykselle, yhden solmun osoittamiselle aliverkon ja solmun osoitteella, sekä solmun sisäiselle kytkennälle. Määrittelyt sisältyvät unioniin ”address_struct”, joka on määritelty Neuron C:n header-tiedostossa ”access.h”.

```
typedef union address_struct {
        group_struct        gp;
        snode_struct        sn;
        bcast_struct        bc;
        turnaround_struct   ta;
} address_struct;
```

Osoitetaulukon merkinnän ensimmäinen tavu ilmaisee aina merkinnän formaatin. Ryhmälähetyksessä merkinnän ensimmäinen bitti on 1. Muissa tapauksissa taulukon formaatti ilmaistaan ensimmäisessä tavussa luettelotyypillä ”addr_type”, joka on määritelty header-tiedostossa ”addr_defs.h”.

```
typedef enum addr_type {
        UNBOUND,
        SUBNET_NODE,
        NEURON_ID,
        BROADCAST
} addr_type;
```

Osoitetaulukon merkinnän ensimmäisessä tavussa 0 voi ilmaista sekä käyttämätöntä merkintäpaikkaa (UNBOUND) että solmun sisäistä kytkentää. Nämä erotetaan toisistaan seuraavan tavun perusteella, jossa sisäistä kytkentää merkitään arvolla 1.

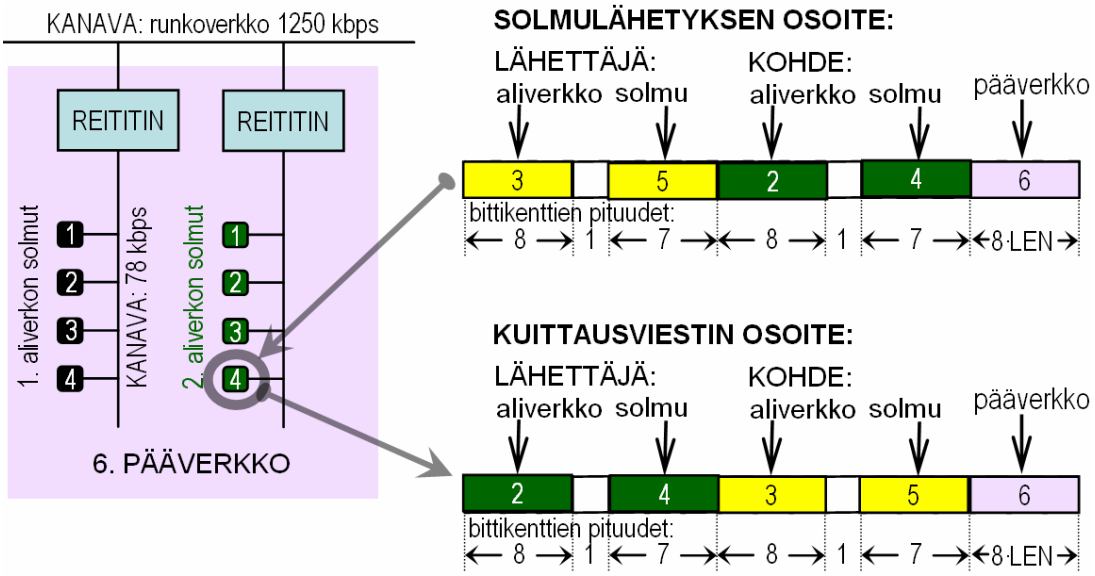
Sovellusohjelma voi kopioida osoitetaulukon merkinnän sovelluksessa määriteltyyn address_struct-tyypin muuttujaan funktiolla ”access_address( )”. Tätä kopiota voidaan muokata ja uudet arvot voidaan päivittää käyttöön funktiolla ”update_address( )” (Echelon 2003a).

Sovelluksessa määritellyn address_struct-tyypin muuttujan arvoja manipuloitaessa tulee merkitä selvästi, mitä unionin alityyppiä uudet arvot edustavat. Tämä tehdään käyttämällä muuttujan nimessä pisteoperaattoria, jolloin pisteen perään merkitään alikentän nimi, esim. ”address_struct_variable.gp”. Alityypin tunnusta tarvitaan muuttujan arvoja asetettaessa, koska tällöin bittikenttien määrittelyt tulee tietää täsmällisesti. Alityyppien bittikenttien samat lukumäärät eivät vielä mahdollista niiden yhdenmukaista käsittelyä, koska toisissa tyypeissä osa kentistä on määritelty anonyymeinä. Ilman nimeä määritelty anonyymi bittikenttä varaa tilan passiivisesti (Miller ym. 1997). Muuttujan sisältämään anonyymiin bittikenttään ei voi kopioida dataa, vaan anonyymi kenttä jätetään kopioinnissa huomiotta.

Osoitetaulukon merkinnät sisältävät verkko-osoitteen lisäksi ajastimet kuittaavan tiedonsiirtopalvelun uudelleenlähetykselle, kahdentuneiden viestien tunnistukseen sekä kuittauksettoman tiedonsiirtopalvelun toistoihin. Myös toistojen lukumäärä määritellään. Nämä arvot voidaan määritellä erikseen jokaiselle verkkoyhteydelle, jolloin saman osoitetaulukon merkinnän käyttö tosin estyy, vaikka kohteen verkko-osoite olisi sama.

6.3.1 Solmulähetys

LonTalk-protokolla käyttää viestipakettien kohdeosoitteena tyypillisesti loogista osoitetta, joka muodostuu pääverkon ja aliverkon osoitteesta sekä solmun omasta tunnuksesta aliverkon sisällä. Kuten kuvasta 7. nähdään, sisältää viestipaketin osoite myös lähettäjän loogisen osoitteen, johon mahdollinen kuittausviesti voidaan palauttaa. Solmu kykenee kuittausviestin lähetykseen ilman osoitetaulukkoon tallennettua kohdeosoitetta, kun se sijoittaa kuittausviestin kohdeosoitteeksi alkuperäisen lähettäjän osoitteen.



Kuva 7. Osoitteiden bittikentät solmulähetyksen viestipaketeissa (Anyong 2000).

Solmulähetyksen osoitetiedot tallennetaan Neuron-piirin EEPROM-muistiin rakenteeseen ”snode_struct”, joka määritellään Neuron C -kielen header-tiedostosta ”addr_defs.h”.

```
typedef struct snode_struct {
        addr_type type;                 // SUBNET_NODE = 1
        unsigned domain      : 1;       // domain index
        unsigned node        : 7;       // node number
        unsigned rpt_timer   : 4;       // unackd_rpt timer
        unsigned retry       : 4;       // retry count
        unsigned             : 4;       // anonymous bit field
        unsigned tx_timer    : 4;       // transmit timer index
        unsigned subnet;                // subnet ID
} snode_struct;
```

Yhden solmun osoituksessa ensimmäinen tavu sisältää osoitetyypin tunnuksen. Sen lukuarvona on 1, mikä vastaa luettelotyypin ” addr_type” arvoa ”SUBNET_NODE”.

Lähtevän viestipaketin tulee sisältää pääverkon osoite ja lähettäjän aliverkon ja solmun tunnukset. Toisen tavun ensimmäinen bitti ilmaisee, kummasta verkkoaluetaulukon merkinnästä nämä tiedot luetaan. Loput tavun biteistä ilmaisevat kohdesolmun loogisen tunnuksen aliverkon sisällä. Eri tunnuksia voi olla 127 kappaletta, koska 0 ei ole sallittu arvo.

Bittikenttä ”rpt_timer” ilmaisee toistavan kuittauksettoman tiedonsiirtopalvelun toistojen välisen ajan. Aika-arvo on koodattu välille 16 ms - 3 s (Toshiba 2006). Toistojen lukumäärä ilmaistaan kentässä ”retry”, jolloin lähetettyjen viestien maksimimäärä on yhden suurempi kuin kentän arvo. Tätä toistojen lukuarvoa sovelletaan toistavaan ja kuittaavaan viestipalveluun sekä sovellusviestien kysely/vastauspalveluun. Näistä kahdessa jälkimmäisessä toistoja suoritetaan vain tarvittaessa, mikäli lähettäjä ei saa palautetta kohteelta.

Solmulähetyksen taulukkomerkinnässä vastaanottoajastimen bittikenttä on määritelty anonyymiksi, eikä siihen voida kopioida dataa. Vastaanottoajastimen arvo luetaan tällöin kokoonpanon parametrien kentästä ”non_group_timer” (Toshiba 2006). Solmu

säilyttää vastaanottamansa viestin tunnisteosaa asetetun ajan, jotta se kykenee suodattamaan toistetut identtiset viestit. Jo saapuneen viestin tunnisteosaa verrataan saapuviin uusiin viesteihin, ja tällöin heijastumien sekä toistavan tiedonsiirtopalvelun aiheuttamat viestien kaksoiskappaleet voidaan jättää huomiotta.

Bittikenttä ”tx_timer” asettaa kuittaavan tiedonsiirtopalvelun ajastimen. Mikäli viestin lähettänyt solmu ei vastaanota kuittausta joltakin kohdesolmuilta ennen ajastimen raukeamista, lähettää se viestin uudelleen. Kentän aika-arvo on koodattu välille 16 ms - 3 s (Toshiba 2006). Ajastimen arvoa asetettaessa on huomioitava verkon topologia, sillä suuren kulkuaikaviiveen omaavassa verkossa ajastin saattaa raueta liian aikaisin, mikä aiheuttaa tarpeettomia uudelleenlähetyksiä ja kuittauksia. Myös sovellusviestien kysely/vastauspalvelu käyttää tätä ajastinta. Tämän palvelumuodon yhteydessä kysyvän solmun tulee huomioida viive, jonka vastaavan solmun sovellusohjelma tarvitsee vastausviestin muodostamiseen.

Solmulähetyksen taulukkomerkinnän viimeinen tavu sisältää aliverkon tunnuksen, yksi pääverkko voi siis sisältää kahdeksan bitin osoittamat 255 aliverkkoa, kun 0 on varattu koko pääverkon kattavalle yleislähetykselle.

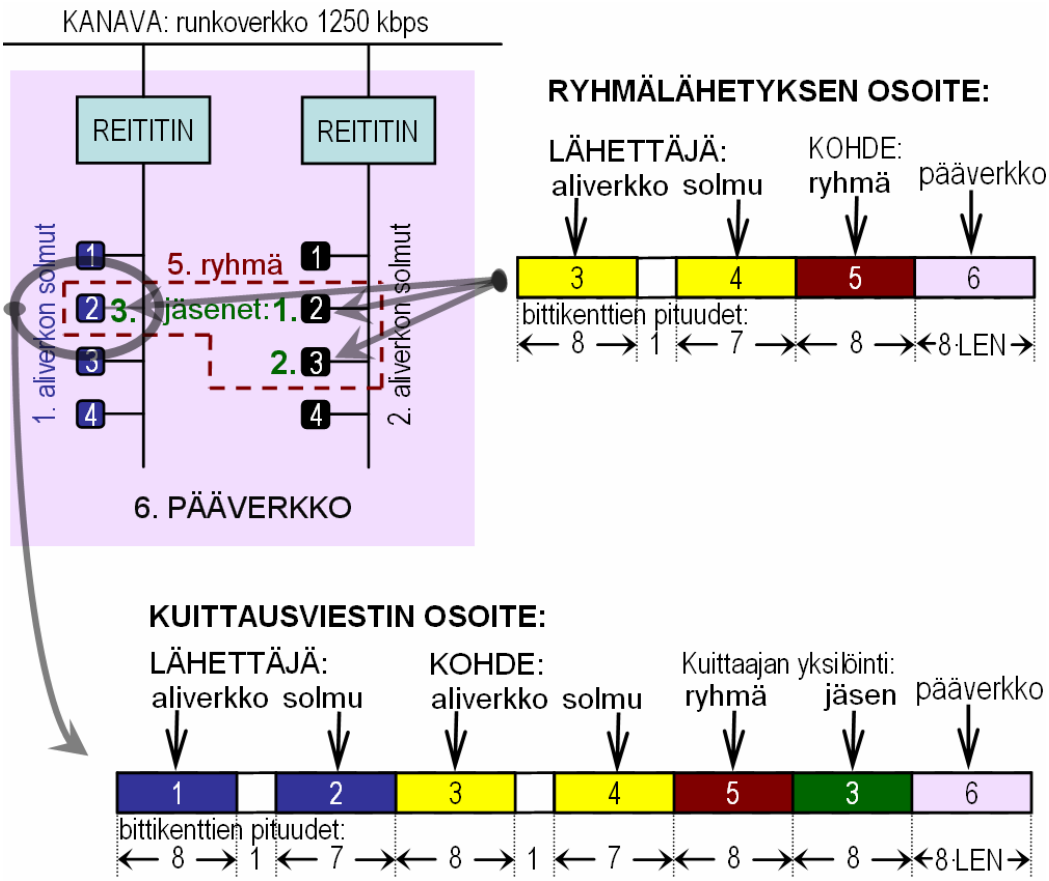
6.3.2 Ryhmälähetys

Mikäli solmujen välisessä loogisessa kytkennässä on enemmän kuin kaksi osapuolta, määritellään niiden välinen kytkentä omaksi ryhmäkseen. Ryhmän kokoonpano määritellään tyypillisesti verkonhallintaohjelmalla solmujen välisiä loogisia kytkentöjä muodostettaessa. Kuvan 8. sisältämä Lon-verkon rakennekuva selventää, kuinka saman ryhmän jäsenet voivat kuulua eri kanaviin ja aliverkkoihin.

Kuva 8. esittää myös osoitteiden rakennetta ryhmälähetyksessä ja sen käynnistämissä kuittausviesteissä. Ryhmälähetysviestissä kohde voidaan osoittaa pelkällä ryhmän tunnuksella. Lisäksi viesti sisältää lähettäjän solmuosoitteen, johon mahdollinen kuittausviesti lähetetään.

Kuittausviestin kohdeosoitteeksi valitaan siis alkuperäisen lähettäjän solmuosoite. Viestiin liitetään kuittaajan solmuosoite, johon mahdolliset lisäkyselyt voidaan helposti lähettää. Jotta kuittaaja voidaan yksilöidä ryhmän jäsenten joukosta, lisätään kuittausviestin osoitteeseen ryhmä- ja jäsennumerot.

Ryhmälähetyksessä kaikki ryhmän sisääntulomuuttujat päivittyvät, kun mikä hyvänsä ryhmän ulostulomuuttuja suorittaa lähetyksen. Kuvaan on merkitty vain yhden solmun lähettämän kuittausviestin osoite, vaikka kuittaavaa palvelua käytettäessä kaikki ryhmän jäsenet suorittavat kuittauksen. Kuva havainnollistaa myös kuittausviestin osoitteen pituutta alkuperäiseen ryhmälähetyksen osoitteeseen verrattuna. Ainoastaan kuittausviestissä tarvitaan jäsennumeroa, jonka perusteella alkuperäinen lähettäjä pitää kirjaa kuittauksen palauttaneista solmuista. Kuittaava tiedonsiirto aiheuttaa siis runsaasti verkkoliikennettä useamman solmun välisessä kytkennässä. Kun lisäksi verkkoliikenteen sisääntulopuskurissa tarvitaan tilaa kaikille ryhmän jäsenten samanaikaisesti lähettämille kuittauksille, on toistava tiedonsiirtopalvelu kuittaavaa parempi suurissa kytkennöissä.



Kuva 8. Osoitteiden bittikentät ryhmälähetyksen viestipaketeissa (Anyong 2000).

Ryhmälähetyksen osoitetiedot tallennetaan Neuron-piirin EEPROM-muistiin rakenteeseen ”group_struct”, joka määritellään Neuron C -kielen header-tiedostosta ”addr_defs.h”.

```
typedef struct group_struct {
        unsigned type       : 1;    // 1 => group
        unsigned size       : 7;    // group size (0 => huge group)
        unsigned domain     : 1;    // domain index
        unsigned member     : 7;    // member num (if huge group, use 0)
        unsigned rpt_timer  : 4;    // unackd_rpt timer
        unsigned retry      : 4;    // retry count
        unsigned rcv_timer  : 4;    // receive timer index
        unsigned tx_timer   : 4;    // transmit timer index
        unsigned group;             // group ID
} group_struct;
```

Ryhmälähetyksen taulukkomerkinnän ensimmäisessä tavussa määritellään osoitetyyppi sekä ryhmän koko lähettäjä mukaan lukien. LonTalk-protokollassa ryhmän koko on rajoitettu 64:ään, ellei käytetä toistavaa tai kertaviestipalvelua. Tällöin ryhmän kokoa ei rajoiteta, mikä ilmaistaan kokomerkinnän arvolla 0.

Toisen tavun ensimmäinen bitti kertoo, kumpaa verkkoaluetaulukon merkintää käytetään. Loput tavun biteistä ilmaisevat solmun jäsennumeron. Jokaiselle ryhmän jäsenelle annetaan oma jäsennumero ryhmää muodostettaessa, paitsi rajoittamattoman ryhmäkoon yhteydessä käytetään tunnusta 0 kaikille ryhmän jäsenille. Kuittaavassa tiedonsiirtopalvelussa lähettäjä pitää kirjaa kuittauksen

palauttaneista solmuista jäsennumeroiden perusteella, ja ainoastaan kuittausviestissä jäsennumero liitetään paketin osoitetietoihin.

Ajastimet määritellään muuten samoin kuin solmulähetyksessä, paitsi vastaanottoajastimen bittikenttä on nyt aktiivisesti käytössä ja ajastimen arvo voidaan näin sovittaa jokaiselle ryhmälle erikseen. Se määritellään bittikentässä “rcv_timer” ja aika-arvo on koodattu välille 128 ms - 24,6 s (Toshiba 2006). Ryhmälähetyksessä kuittaavan tiedonsiirtopalvelun ajastin ”tx_timer” asetetaan uuden lähetysyrityksen lisäksi myös vastaanotettaessa kuittaus joltakin ryhmän jäseneltä viimeistä kuittausta lukuun ottamatta.

Viimeinen ryhmälähetyksen osoitetaulukon tavu sisältää ryhmän tunnuksen, yhdessä pääverkossa voi siis olla 256 ryhmää. Koska tiedot verkkomuuttujan ryhmä- ja jäsennumerosta tallennetaan ainoastaan osoitetaulukon merkintään, tarvitaan se myös ryhmään kytketylle sisääntulomuuttujalle.

6.3.3 Yleislähetys

Yleislähetyksessä viestin vastaanottajina toimivat kaikki pääverkon tai tietyn aliverkon solmut. Echelonin tuottamat verkonhallintaohjelmat eivät käytä yleislähetystä verkkomuuttujien tai sovellusviestien välisissä loogisissa kytkennöissä. LonTalk-protokolla kuitenkin tukee tätä lähetysmuotoa, ja sitä voidaan käyttää eksplisiittisen osoitemuodon yhteydessä (Toshiba 2006). Verkonhallintatyökalu voi esim. tiedustella tiedonsiirtokanavaan kytkettyjen solmujen ID-tunnuksia yleislähetystä käyttävän verkonhallintaviestin avulla.

Yleislähetyksen osoitetaulukon rakenne on lähes samanlainen kuin solmulähetyksessä:

```
typedef struct bcast_struct {
        addr_type type;                 // BROADCAST = 3
        unsigned domain       : 1;      // domain index
        unsigned              : 1;
        unsigned backlog      : 6;      // backlog override value
        unsigned rpt_timer    : 4;      // unackd_rpt timer
        unsigned retry        : 4;      // retry count
        unsigned              : 4;
        unsigned tx_timer     : 4;      // transmit timer index
        unsigned subnet;                // subnet ID (0 => domain broadcast)
} bcast_struct;
```

Yleislähetyksen osoitetaulukon ensimmäinen tavu ilmaisee osoitetyypin tunnuksen. Yleislähetykselle on assosioitu luettelotyypin ”addr_type” arvo ”BROADCAST”, joka vastaa lukuarvoa 3. Toisen tavun ensimmäinen bitti ilmaisee verkkoaluetaulukon indeksin ja kuusi viimeistä sisältävät uuden verkonkuormituskertoimen yleislähetyksen vastaanottaneille solmuille. Tällöin ne arpovat väylälle liittymishetkensä aiempaa suuremmasta aikaikkunoiden joukosta ja törmäysten määrä jää mahdollisimman pieneksi. Näin voidaan varautua verkon kuormituksen hetkelliseen lisäykseen, mikä aiheutuu yleislähetyksen vastaanottaneiden solmujen mahdollisista kuittausviesteistä. Perusoikeuksilla varustettu solmu arpoo verkkoon liittymishetkensä vähintään 16 aikaikkunan väliltä. Tämä luku kerrotaan kuormituskertoimella (backlog), jota merkitään kuudella bitillä ja se saa arvoja väliltä 1 - 63. Törmäysten välttäminen verkon kuormitusta ennustamalla ja mahdollisten aikaikkunoiden lukumäärää muuttamalla muodostaa perustan LonTalk-protokollan kilpavarausperiaatteelle, predictive-persistent CSMA:lle. Kuormituskertoimeksi tulisi

sijoittaa arvio yleislähetyksen aiheuttamien kuittausviestien lukumäärästä. Mikäli tätä ei tiedetä, käytetään ”backlog”-kentän arvolla 0 kuormituskertoimen oletusarvoa 15.

Kuittaavan tiedonsiirtopalvelun ajastinta asetettaessa tulee huomioida, että yleislähetyksessä ensimmäinen alkuperäisen lähettäjän vastaanottama kuittaus päättää tiedonsiirtotapahtuman, ja myöhemmät kuittaukset jätetään huomiotta. Vastaava ominaisuus on myös sovellusviestien kysely/vastauspalvelulla.

Viimeinen tavu sisältää jälleen aliverkon tunnuksen, jossa arvolla 0 voidaan ilmaista koko pääverkon kattava yleislähetys.

6.3.4 Eksplisiittinen osoite

Verkkomuuttujan päivitysviestin tai sovellusviestin kohdeosoite voidaan muodostaa sovellusohjelmassa. Tällöin sovellusohjelman kirjoittaja voi täsmällisesti valita osoitteen rakenteen ja täyttää valitsemansa osoitetyypin kentät. Tällaista eksplisiittistä kohdeosoitetta ei tallenneta EEPROM-muistin osoitetaulukkoon, vaan se muodostetaan aina sovellusohjelmassa. Eksplisiittistä osoitetta käytettäessä on myös verkkomuuttujan päivitysviesti muodostettava sovellusviestin mekanismeja käyttäen. Tällöin verkkomuuttujien automaattisia palveluja jää hyödyntämättä, mutta viestin osoite voidaan määritellä täsmällisesti sovellusohjelmassa.

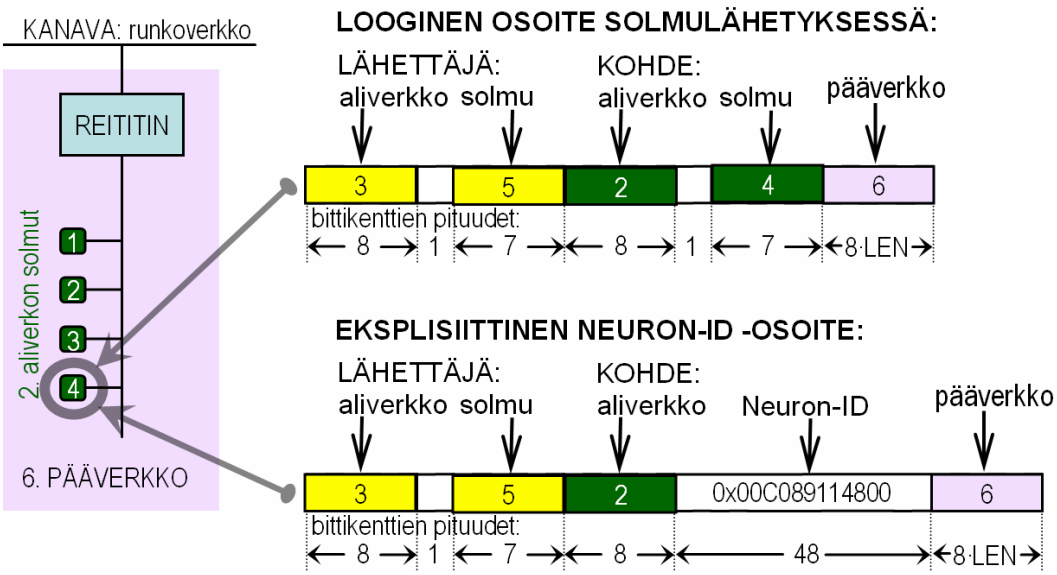
Lähetettävä sovellusviesti muodostetaan Neuron C:ssä valmiiksi määriteltyyn viestiobjektiin ”msg_out” ja viesti lähetetään kutsumalla funktiota ”msg_send( )”. Määriteltäessä kohdeosoite eksplisiittisesti täytetään osoitetiedot viestiobjektin muuttujaan ”dest_addr”. Tämä muuttuja on unioni, jonka alityyppi valitaan käyttöön muuttujan nimessä käytettävällä pisteoperaattorilla, esim. ”msg_out.dest_addr.snode”.

Muuttujan ”dest_addr” tyyppimäärittely on unionin ”msg_out_addr” mukainen. Määrittely löytyy otsikkotiedostosta ”msg_addr.h”.

```
typedef union msg_out_addr {
        addr_type               no_address; // UNBOUND (i.e. 0) if no address
        msg_out_group_struct group;
        snode_struct            snode;
        nrnid_struct            nrnid;
        bcast_struct            bcast;
        addrt_struct            addrt;
} msg_out_addr;
```

Määrittelystä nähdään, että eksplisiittinen osoite voidaan määritellä ryhmälle (group), yhdelle solmulle sekä loogisen (snode) että fyysisen (nrid) osoitteen perusteella sekä kaikille pää- tai aliverkon solmuille (bcast). Lisäksi eksplisiittinen osoite voi käyttää EEPROM-muistin osoitetaulukon merkintää (addrt).

Eksplisiittisesti solmua osoitettaessa voidaan osoitteena käyttää Neuron-piirin 48-bittistä ID-tunnusta. Kuvan 9. mukaisesti osoitteen rakenne muistuttaa tällöin solmulähetyksen loogista osoitetta. Eksplisiittisessä osoitteessa ainoastaan korvataan solmun looginen tunnus Neuron-piirin ID-tunnuksella. Tällöin aliverkon osoitetta käytetään viestin reitittämiseen, ja viestin tulisi läpäistä kaikki pääverkon reitittimet aliverkon osoitteen arvolla 0. Asentamaton solmu voi lisäksi vastaanottaa ID-tunnuksellaan varustetun viestin missä verkkoalueessa tahansa. Tällöin se muodostaa mahdollisen vastausviestin osoitteen vastaanottamansa viestin lähdetietojen perusteella.



Kuva 9. Loogisen ja eksplisiittisen Neuron-ID -osoitteen vertailu (Anyong 2000).

Eksplisiittiseen Neuron-ID -osoitteeseen valitaan unionin ”dest_addr” alityypiksi ”nrnid”, jonka rakenne on määritelty tiedostossa msg_addr.h. Tätä rakennetta ei siis säilytetä EEPROM-muistin osoitetaulukossa vaan kentät täytetään sovellusohjelmassa.

```
typedef struct nrnid_struct {
        addr_type       type;           // NEURON_ID
        unsigned        domain: 1;
        unsigned                        : 7;
        unsigned        rpt_timer       : 4;
        unsigned        retry           : 4;
        unsigned                        : 4;
        unsigned        tx_timer        : 4;
        unsigned        subnet;
        unsigned        nid[NEURON_ID_LEN]; // NEURON_ID_LEN = 6
} nrnid_struct;
```

Eksplisiittisen Neuron-ID -osoitteen muodostava rakenne muistuttaa solmulähetyksen osoitetaulukon merkintää, solmun loogisen tunnuksen kenttä on vain jätetty anonyymiksi ja loppuun on lisätty kenttä ID-tunnukselle.

Neuron-piirin ID-tunnuksen käyttö eksplisiittisessä osoitteessa lisää viestipaketin osoitteen pituutta tuntuvasti. Se on siis vain yksi vaihtoehtoinen tapa muodostaa eksplisiittinen osoite, ja myös loogista solmun, ryhmän ja yleislähetyksen osoitetta voidaan käyttää.

Eksplisiittistä osoitetta käytettäessä osoitteen muodostus siirtyy verkkokerrokselta sovellusohjelmalle, jolloin verkkoliikenteen sovelluspuskuri sisältää 11-tavuisen osoitekentän (Toshiba 2006). Sovellus- ja verkkopuskureille suositellaan tällöin samaa kokoa, eivätkä sovelluspuskurit voi olla verkkopuskureita pienempiä kuten tavallisesti (Echelon 2003b).

EEPROM-muistin osoitetaulukkoon mahtuu merkinnät vain 15 kohdeosoitteelle. Eksplisiittistä osoitetta käytettäessä tämä rajoite poistuu ja yksilöllisiä kohdeosoitteita voidaan määritellä sovellusohjelmassa runsaasti.

Eksplisiittistä osoitetta voidaan hyödyntää myös useampaa identtistä laitetta tarkkailevassa valvontalaitteessa. Esim. hälytysjärjestelmän näyttölaitteeseen voidaan kytkeä monta hälytyksen laukaisevaa anturia, jotka kytketään samaan sisääntulomuuttujaan valvontalaitteessa. Sisääntulon muuttuessa tulee valvontalaiteen kyetä yksilöimään hälytyksen suorittanut laite. Tämä onnistuu, kun antureiden ja valvontalaitteen välinen kytkentä käyttää kyselypalvelua ja eksplisiittistä osoitetta. Tällöin valvontalaite pyytää sisääntulomuuttujansa päivitystä yhdeltä anturilta kerrallaan ja voi näin yksilöidä päivityksen suorittaneen laitteen.

Verkon kokoonpanon muuttuessa on solmujen sovellusohjelmissa määriteltyjen eksplisiittisten osoitteiden muuttaminen työlästä. Muutoksia ei voida tehdä suoraan verkonhallintaviesteillä kunkin solmun EEPROM-muistin asetustaulukoihin. Itseasentuvat laitteet on suunniteltu käytettäväksi ilman verkonhallintaviestejä, sillä ne kykenevät vastaanottamaan uudet osoitetiedot paikallisen syötteen avulla esim. I/O-rajapinnan välityksellä. Eksplisiittisten osoitteiden käyttö onkin yleistä juuri itseasentuvissa laitteissa, joiden ylläpidettävyys ei huonone tämän osoitemuodon myötä.

6.4 Verkkomuuttujataulukot

Verkkomuuttujien asetustiedot sijaitsevat kolmessa taulukossa. Kiinteä taulukko on osa sovellusohjelmaa ja sisältää ohjelmointityökalun linkittäjän asettamat verkkomuuttujien muistiosoitteet RAM-muistiin. Verkkomuuttujien varsinainen asetustaulukko sisältää puolestaan tiedot verkkomuuttujan käyttämistä tiedonsiirron palveluista, sen loogisen kytkennän tunnuksena toimivan valitsimen sekä osoitetaulukon indeksin muuttujan käyttämän osoitemerkinnän valitsemiseksi. Aliastaulukon avulla asetustaulukon verkkomuuttujaan voidaan assosioida ylimääräisiä verkkomuuttujan tunnuksena toimivia valitsimia. Tämä mahdollistaa monipuolisemmat kytkennät solmua verkkoon asennettaessa.

Sekä kiinteän että varsinaisen verkkomuuttujien asetustaulukon merkinnöille varataan tilaa sovellusohjelman tarpeiden mukaisesti sen kääntämisen yhteydessä. Itsenäisesti toimivassa Neuron-piirissä näiden taulukoiden koko on rajoitettu 62 merkintään, mikä on yhden solmun verkkomuuttujien suurin mahdollinen lukumäärä. Muuttujavektorin jokainen elementti vaatii oman merkintäpaikan taulukoihin. Mikäli Neuron-piiri kytketään erilliseen isäntäprosessoriin, sisältää isäntäprosessorin muisti verkkomuuttujat ja kiinteän taulukon sekä mahdollisesti myös asetus- ja aliastaulukon. Tällöin verkkomuuttujien ja aliaksien enimmäismäärät ovat molemmille erikseen 4096, ja verkkomuuttujien päivitystehtävä siirtyy isäntäprosessorille (Toshiba 2006).

6.4.1 Kiinteä verkkomuuttujataulukko

Kiinteä verkkomuuttujataulukko sisältää verkkomuuttujien muistiosoitteet RAM-muistiin, jotka ohjelmointityökalun linkittäjä on asettanut sovellusohjelman kääntämisen yhteydessä. Kiinteä taulukko on osa sovellusohjelmaa, ja se voi sijaita erillisellä muistipiirillä Neuron 3150 -kontrolleria käytettäessä. Taulukon arvojen asettamiseen ei ole omia varusohjelmiston funktioita tai verkonhallintaviestejä, ja muutokset siihen tehdään lataamalla uusi sovellusohjelma. Sovelluskohtaiset parametrit sisältävät tosin kiinteään taulukon muistiosoitteen, mikä on luettavissa globaalin muuttujan kentästä ”read_only_data.nv_fixed”. Kenttä ”read_only_data.nv_count” sisältää verkkomuuttujien lukumäärän ja jokainen kiinteän taulukon merkintä vie kolme tavua muistia. Näin taulukon sijainti on täsmällisesti määritelty ja sen sisältö on luettavissa verkonhallintaohjelmalla. Kiinteän verkkomuuttujataulukon merkintä kuvataan rakenteessa ”nv_fixed_struct”, joka määritellään Neuron C -kielen header-tiedostosta ”access.h”.

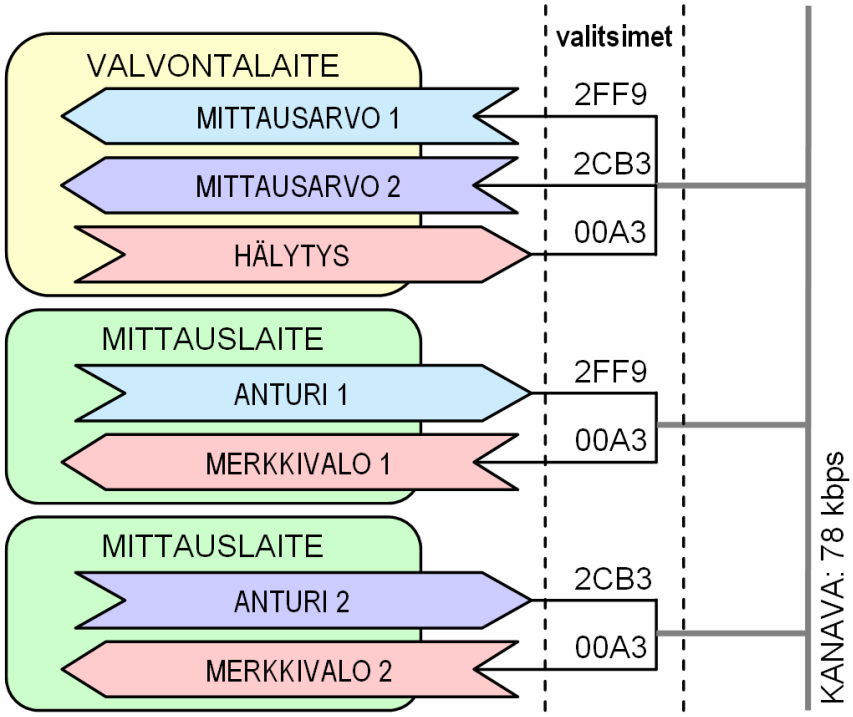
```
typedef struct nv_fixed_struct {
        unsigned        nv_sync         : 1;
        unsigned                        : 2;
        unsigned        nv_length       : 5;
        void *          nv_address;
} nv_fixed_struct;
```

Kiinteän taulukon merkinnän ensimmäinen bitti ilmaisee, onko verkkomuuttuja määritelty synkroniseksi. Mikäli ulostuloverkkomuuttuja on synkroninen, lähetetään jokainen sen saama arvo eteenpäin, vaikka arvo päivittyisi nopeammin kuin niitä ehditään verkkoon syöttämään. Tällöin lähtevän datan puskuri saattaa täyttyä jolloin sovellusohjelma joutuu odottamaan puskurin vapautumista. Jokaisen päivitystiedon välittäminen on välttämätöntä käytettäessä suhteellista mittausperiaatetta, jolloin viestit sisältävät vain muutosarvoja. Muulloin asynkroninen toiminta on tehokkaampaa, kun lähetetään vain tieto viimeisimmästä arvosta. Sovellusohjelmassa muuttuja asetetaan synkroniseksi määrittelyn yhteydessä määreellä ”sync”, esim. ”network output sync SNVT_alarm nvoAlarm” (Echelon 2003b).

Bittikenttä ”nv_length” kertoo verkkomuuttujan pituuden tavuina, jolloin suurin pituus on 31 tavua. Rakenteessa viimeisenä oleva verkkomuuttujan muistiosoite vie kaksi tavua, jolloin yhden verkkomuuttujan merkintä vaatii kolme tavua muistitilaa kiinteässä taulukossa.

6.4.2 Verkkomuuttujien asetustaulukko

Verkkomuuttujien varsinainen asetustaulukko sijaitsee EEPROM-muistissa, ja sen tietoja päivitetään aina Lon-verkon solmujen välisiä loogisia kytkentöjä muutettaessa. Asetustaulukko sisältää tiedot verkkomuuttujan käyttämistä tiedonsiirron palveluista sekä osoitetaulukon indeksin muuttujan käyttämän osoitemerkinnän valitsemiseksi. Lisäksi asetustaulukko määrittelee verkkomuuttujien väliset kytkennät valitsinten avulla.



Kuva 10. Kytkettäessä solmut loogisesti luodaan verkkomuuttujille valitsimet.

Verkkomuuttujan valitsin on sama kaikissa loogisesti yhteen kytketyissä verkkomuuttujissa. Tyypillisesti verkonhallintatyökalu luo valitsimet solmuja yhdistettäessä. Kun Lon-verkon solmu vastaanottaa verkkomuuttujan päivitysviestin, se osaa sijoittaa viestin sisältämän arvon oikeaan sisääntuloverkkomuuttujaan juuri valitsimen perusteella. Kuvan 10. esimerkissä valvontalaite vastaanottaa kahden mittauslaitteen antureiden lukemat omiin sisääntuloverkkomuuttujiinsa. Lisäksi valvontalaite voi suorittaa hälytyksen, joka sytyttää merkkivalon kummassakin mittauslaitteessa.

Asetustaulukon yhden merkinnän tiedot kuvataan rakenteessa ”nv_struct”, joka määritellään Neuron C -kielen header-tiedostossa ”access.h”.

```
typedef struct nv_struct {  // nv_struct is returned by the function access_nv( )
        unsigned nv_priority        : 1;
        unsigned nv_direction       : 1;    // 1 => output
        unsigned nv_selector_hi     : 6;
        unsigned nv_selector_lo     : 8;
        unsigned nv_turnaround      : 1;
        unsigned nv_service         : 2;
        unsigned nv_auth            : 1;
        unsigned nv_addr_index      : 4;    // Address table index
} nv_struct;
```

Asetustaulukon ensimmäinen bitti ilmaisee verkkomuuttujan mahdollisen prioriteettimäärityksen. Prioriteettiverkkomuuttujan lähettämiselle on varattu oma aikaikkuna, ja sillä on ensimmäisenä mahdollisuus lähetykseen aina jokaisen lähetetyn paketin jälkeen. Prioriteetti voidaan asettaa sovellusohjelmassa verkkomuuttujan määrittelyn yhteydessä määreellä ”bind_info (priority)”.

Seuraava bitti ilmaisee verkkomuuttujan suunnan, ykkösen merkitessä ulostuloa. Sen arvoa ei ole tarkoitettu muutettavaksi verkkoasetuksia tehtäessä, rakenteen tehokkuus on vain edellyttänyt sen sijoittamista asetustaulukkoon (Toshiba 2006).

Verkkomuuttujan valitsimelle on varattu bittikentät ”nv_selector_hi” ja ”nv_selector_low”. Näillä yhteensä 14 bitillä voidaan ilmaista arvot välillä 0 - 0x3FFF. Valitsinten arvot 0x3000 - 0x3FFF on varattu kytkemättömille verkkomuuttujille, jolloin arvoksi tulee 0x3FFF vähennettynä muuttujan indeksillä. Tätä pienemmät valitsinten arvot ovat käytettävissä kytketyille verkkomuuttujille. Varusohjelma sisältää funktiot high_byte( ) ja low_byte( ), jolla voidaan erottaa kaksitavuisen luvun ylä- ja alabitit erillisiksi tavuiksi. Nämä voidaan syöttää asetustaulukon bittikenttiin ”nv_selector_hi” ja ”nv_selector_low”.

Bitti ”nv_turnaround” ilmaisee, jos looginen kytkentä on tehty solmun sisällä eli saman solmun sisään- ja ulostulo on yhdistetty. Bittikentällä ”nv_service” merkitään kuljetuskerroksen tarjoamaa tiedonsiirron palvelumuotoa. Mahdollisia ovat kuittaava, toistava sekä kertaviesteihin pohjautuva yhteyspalvelu, joiden tunnuskoodeina toimivat luvut 0 - 2. Oletuksena käytetään kuittauspalvelua. Sovellusohjelmassa voidaan asettaa toistava palvelu verkkomuuttujan määrittelyn yhteydessä määreellä ”bind_info (unackd_rpt)”, sekä kertaviestipalvelu määreellä ”bind_info (unackd)”.

Bitti ”nv_auth” saa arvon yksi, mikäli verkkomuuttujan päivitysviestin siirrossa käytetään aidonnuspalvelua. Sovellusohjelmassa se asetetaan verkkomuuttujan määrittelyssä määreellä ”bind_info (authenticated)”. Viimeinen puolitavu on varattu osoitetaulukon indeksille, joka kertoo mihin osoitetaulukon merkintään kyseinen

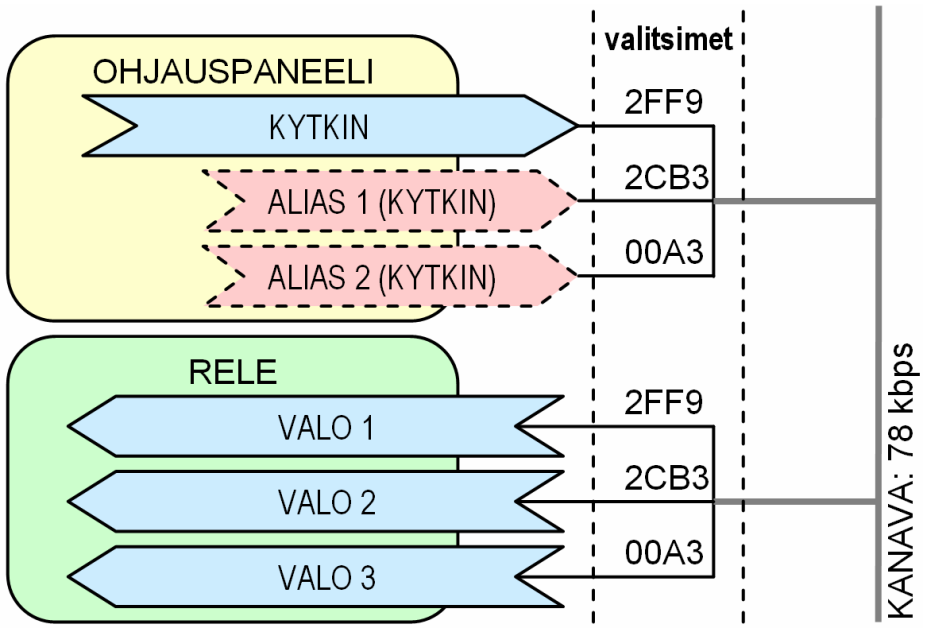
verkkomuuttuja on assosioitu. Osoitetaulukkoon on oletuksena varattu paikkoja 15:lle eli maksimimäärälle merkintöjä, jolloin indeksit saavat arvoja väliltä 0 - 14. Indeksin arvo 15 ilmaisee, ettei verkkomuuttujaa ole assosioitu osoitetaulukkoon. Useampi verkkomuuttuja voi käyttää samaa osoitetaulukon merkintää, mikäli niiden kohdeosoite sekä protokollan ajastusarvot ovat samoja.

Sovellusohjelma voi kopioida asetustaulukon merkinnän sovelluksessa määriteltyyn nv_struct-tyypin muuttujaan funktiolla ”access_nv( )”. Tätä kopiota voidaan muokata ja uudet arvot voidaan päivittää käyttöön funktiolla ”update_nv( )” (Echelon 2003a). Sekä lukevaan että päivittävään funktioon voidaan antaa argumenttina funktio ”nv_table_index( )”, joka palauttaa verkkomuuttujan indeksin asetustaulukkoon, esim. ”access_nv(nv_table_index(nvoSwitch))”. Sekä asetustaulukon että kiinteän taulukon indeksit määräytyvät verkkomuuttujien määrittelyjärjestyksestä sovellusohjelmassa.

6.4.3 Aliastaulukko

Alias on verkkomuuttujan abstraktio, joka helpottaa verkkomuuttujien kytkemistä Lon-verkon solmujen välillä. Aliakset mahdollistavat kytkentöjä, jotka eivät ole mahdollisia pelkällä osoitetaulukolla ja verkkomuuttujien asetustaulukolla. Kytkentöjä tekevä verkonhallintaohjelma luo aliakset solmun aliastaulukkoon, ja Neuron-piirin varusohjelmisto huolehtii aliaksien kautta lähetettävistä ja vastaanotettavista viesteistä.

Aliaksien avulla asetustaulukon sisältämille verkkomuuttujille voidaan määritellä ylimääräisiä valitsimia. Näitä tarvitaan, koska saman laitteen sisääntuloverkkomuuttujilla tulee olla eri valitsimet. Lähettävässä laitteessa on käytettävä aliasverkkomuuttujia, kun halutaan samanaikaisesti päivittää useita vastaanottavan laitteen verkkomuuttujia (Echelon 2006). Kuvan 11. esimerkissä ohjauspaneelin kytkimen halutaan ohjaavan kaikkia kolmea releeseen kytkettyä valaisinryhmää, jolloin lähettävään laitteeseen luodaan kaksi alkuperäiseen ulostuloverkkomuuttujaan assosioitua aliastunnusta.



Kuva 11. Aliakset tarjoavat ylimääräisiä valitsimia ulostuloverkkomuuttujalle.

Alias perii alkuperäisen verkkomuuttujan arvon, tyypin ja suunnan. Kun sovellusohjelma päivittää solmun oman ulosmenevän verkkomuuttujan arvoa, lähettää varusohjelmisto verkkomuuttujien päivitysviestit sekä alkuperäisestä että

kaikista siihen liitetyistä aliasverkkomuuttujista. Varusohjelmisto reagoi ulostulon aliakselle osoitettuun päivityskyselyyn lähettämällä verkkomuuttujan päivitysviestissä alkuperäisen verkkomuuttujan arvon.

Myös sisääntuloverkkomuuttujalle voidaan luoda aliaksia. Sisääntulon aliaksen vastaanottaessa verkkomuuttujan päivitysviestin päivittää varusohjelmisto alkuperäisen verkkomuuttujan arvon ja luo sitä vastaavan tapahtumatiedon sovellusohjelman tehtävien ajoittamiseksi. Mikäli sovellusohjelma pyytää sisääntuloverkkomuuttujansa päivitystä, lähettää varusohjelmisto päivityskyselyt sekä alkuperäiseen sisääntulomuuttujaan että sen aliaksiin assosioituihin kohdeosoitteisiin (Motorola 1997b).

Aliastaulukon yhden merkinnän tiedot kuvataan rakenteessa ”alias_struct”, joka määritellään Neuron C -kielen header-tiedostossa ”access.h”.

```
typedef struct alias_struct {
        nv_struct       nv_cnfg;
        unsigned        primary;
        unsigned long   host_primary;
} alias_struct;
```

Aliastaulukon merkintä sisältää verkkomuuttujien asetustaulukon tietorakenteen sekä indeksin siihen asetustaulukon verkkomuuttujaan, johon tämä alias on assosioitu. Nelitavuisessa aliasmerkinnässä bittikenttä ”primary” sisältää tämän indeksin.

Aliastaulukon merkinnän laajemmassa määrittelyssä bittikentän “primary” lukuarvona on 0xFF ja indeksimerkintää varten otetaan käyttöön kaksitavuinen kenttä ”host_primary”. Kaksitavuista indeksiä tarvitaan vain, kun käytetyn indeksin lukuarvo on suurempi kuin 0xFE (Toshiba 2006).

Sovellusohjelma voi kopioida aliastaulukon merkinnän sovelluksessa määriteltyyn alias_struct-tyypin muuttujaan funktiolla ”access_alias( )”. Tätä kopiota voidaan muokata ja uudet arvot voidaan päivittää käyttöön funktiolla ”update_alias( )” (Echelon 2003a).

Toisin kuin kiinteän ja varsinaisen verkkomuuttujien asetustaulukon kohdalla, ei aliastaulukon koko määräydy automaattisesti sovellusohjelman tarpeiden perusteella. Aliaksien tarve ilmenee vasta asennuskohteessa, jonka kytkentöjen toteuttaminen saattaa niitä edellyttää. Tämä tarve voidaan ennakoida varaamalla aliastaulukolle kaikki muuten käyttämättä jäävä EEPROM-muisti.

Aliastaulukon koko määritetään sovellusohjelman kääntämisen yhteydessä. Kääntäjä ei sisällä määrittelyä aliastaulukon oletuskoolle, joten sen merkintäpaikkojen lukumäärä tulee aina asettaa sovellusohjelmassa seuraavalla kääntäjän käskyllä:

```
#pragma num_alias_table_entries      n
```

Tässä sallitut arvot ovat välillä 0 - 62 ja jokainen merkintäpaikka vie neljä tavua muistitilaa. Erillisen isäntäprosessorin hoitaessa verkkomuuttujien päivitystehtävää tulee aliaksien enimmäismääräksi 4096 ja yhden merkinnän koko kasvaa tarvittaessa kuuteen tavuun. Kuuden tavun aliasmerkintä mahdollistaa kaksitavuisen indeksin verkkomuuttujien asetustaulukkoon, jolloin indeksin lukuarvo voi olla suurempi kuin 0xFE.

Itsenäisesti toimivassa Neuron-piirissä voi aliastaulukko sijaita vain integroidussa EEPROM-muistissa. Kaikki sen muuten käyttämättömäksi jäävä kapasiteetti suositellaan asetettavaksi aliastaulukon käyttöön, jotta verkkoasetukset voidaan myöhemmin tehdä mahdollisimman joustavasti. Sovellusohjelman kehityksen alkuvaiheeseen on tarjolla aliastaulukon koon laskukaava, jonka tulosta voidaan myöhemmin korjata vapaaksi jäävän EEPROM-muistin määrällä (Echelon 2003b). Kääntämisen yhteydessä vapaaksi jääneen integroidun EEPROM-muistin määrä ilmenee linkittäjän tiedoista. Ulkoista muistia sisältävässä piirissä aliastaulukon kokoa voidaan kasvattaa sovellusohjelman EEPROM-muistissa sijaitsevan osuuden verran, jolloin sovellusohjelma linkittyy kääntämisen yhteydessä ulkoiseen muistiin.

6.5 Echelonin suositukset itseasentuville laitteille

LonWorks-tekniikan kehittänyt Echelon on määritellyt suositukset verkkoasetusten tekemiseen sovellusohjelmasta käsin (Echelon 2006). Tehtäessä verkkoasetukset muuten kuin verkonhallintaohjelmalla, lupaa Echelon käyttäjätukea vain näiden ohjeiden mukaisille laitteille.

Suositusten mukainen laite ei saa käyttää verkonhallintaviestejä muiden laitteiden verkkoasetusten muuttamiseen. Se ei saa suorittaa verkonhallintaa, jossa verkon asetuksia ylläpidetään tietokannassa ja pyritään muuttamaan verkon sisältämien laitteiden asetuksia yhdenmukaisiksi tietokannan kanssa. Echelon tukee verkonhallintaohjelmistojen kehitystä vain, mikäli ne pohjautuvat Echelonin kehittämään Lon-verkon käyttöjärjestelmään nimeltään LNS. Kaikkien verkonhallintatoimintojen tulisi tapahtua LNS:n kautta, joten muiden laitteiden verkkoasetuksia muuttavia itseasentuvia laitteita ei tueta.

Suositusten mukainen itseasentuviin laitteisiin pohjautuva verkko ei saa sisältää reitittimiä ja rajoittuu siksi yhteen kanavaan. Häiriöisessä verkossa signaalia vahvistavia toistimia voidaan kuitenkin käyttää. Mikäli itseasentuvat laitteet suunnitellaan myös verkonhallintatyökaluilla asennettaviksi, ovat itseasentumisen mahdollistavat funktiot kyettävä kytkemään pois konfliktien välttämiseksi verkonhallintasovellusten kanssa. Poiskytkentäominaisuus tulee toteuttaa standardin asetusparametrin ”SCPTnwrkCnfg” avulla.

Suositusten mukainen itseasentuva solmu on kloonilaite, jolla ei ole yksilöllistä loogista osoitetta. Laite asetetaan klooniksi kirjoittamalla verkkoaluetaulukon merkintä funktiolla ”update_clone_domain( )”. Tällöin viestit voidaan vastaanottaa, vaikka lähettäjän osoite on sama kuin vastaanottajan. Kloonilaitteista voidaan muodostaa ryhmiä aliverkko- ja ryhmätunnusten perusteella, jotka asetetaan I/O-portin kautta tulevan paikallisen syötteen perusteella.

Echelonin suositusten tarkoituksena on tehdä selkeä jako itseasentuvien ja verkon välityksellä asennettavien laitteiden välille. Verkonhallintaviestien käyttö halutaan rajata LNS-pohjaisille verkonhallintatyökaluille. Lisäksi niiden häiriötön toiminta halutaan turvata, mikäli verkossa käytetään myös itseasentuvia laitteita.

Suositukset määrittelevät Echelonin itseasentuviin laitteisiin kohdistuvaa politiikkaa. Niiden perusteella voidaan ymmärtää, miksei Echelonin toimittamassa ohjelmointioppaassa (Echelon 2003b) neuvota verkonhallintafunktioiden tai asetustaulukoiden käyttöä. Echelonin hakuteoksesta (Echelon 2003a) löytyy jo apua verkkoasetusten tekemiseen sovellusohjelmasta käsin, mutta siinäkään ei neuvota verkonhallintafunktioiden käyttöä. Echelon tarjosi 1990-luvun alussa ohjeita verkonhallintatoimintojen toteuttamiseksi Neuron-piirin sovellusohjelmissa. Näistä datakirjoista ja tiedotteista löytyy luettelo (Echelon 1993), mutta niitä ei ole enää saatavilla internetistä tai NodeBuilder-ohjelmiston mukana. Yhtiön politiikkana on

ensisijaisesti tukea LNS:n käyttöä eikä verkkoasetusten tekoa haluta jättää sovellusohjelman kirjoittajille. LonWorks-tekniikan käyttäjien kannalta politiikan etuna on verkonhallintasovellusten yhdenmukaistumisen tarjoama yhteensopivuus. Lisäksi useamman sovelluksen käyttäessä LNS:n tarjoamaa verkkoasetusten tietokantaa ovat ajantasaiset tiedot aina kootusti yhdessä paikassa. LNS-tietokanta voi palvella samanaikaisesti useampaa verkon asetuksia käsittelevää työkaluohjelmaa. Toisaalta politiikka tekee LonWorks-tekniikan käyttäjät entistäkin riippuvaisemmiksi Echelonista, ja ne maksavat nyt sille lisenssimaksuja useamman tuotteen kautta.

6.6 Asetustaulukoiden soveltaminen

Neuron-piiri sisältää varusohjelmiston, joka helpottaa ohjelman kirjoittamista LonWorks-ympäristöön. Varusohjelmiston funktiot hoitavat tiedonvälityksen solmusta Lon-verkkoon, ja ohjelmoinnissa voidaan keskittyä solmun paikallisen tehtävän hoitamiseen, esim. mittaukseen tai toimilaitteen säätöön.

Uusilla verkonhallintaohjelmistoilla Lon-verkko voidaan muodostaa helposti yhdistämällä solmut graafisen käyttöliittymän kautta. Verkonhallintaohjelmisto muodostaa tällöin tietokannan, jonka avulla verkon asetuksia voidaan kätevästi myöhemminkin muuttaa. Neuron-sovellusta ohjelmoitaessa tai Lon-verkkoa muodostettaessa ei siis ole aina välttämätöntä tuntea LonTalk-protokollan toteutuksen yksityiskohtia. Niiden tunteminen mahdollistaa kuitenkin verkon suorituskyvyn optimoinnin, ongelmakohtien löytämisen sekä valmistajakohtaiset räätälöidyt ratkaisut. Lisäksi menetettäessä ylläpidettävän verkon tietokanta ja muu dokumentaatio, voidaan koko verkon tiedonsiirron asetukset silti jäljittää lukemalla yksittäisten solmujen asetustaulukot (Apte Consulting 1998).

Tässä diplomityössä valmistellun työkurssin laitteisto muodostaa hajautetun hissinohjausjärjestelmän simulointiympäristön. Sen sisältämä LonWorks-tekniikka edustaa räätälöityä valmistajakohtaista erityisratkaisua, joka poikkeaa eri valmistajien kesken yhteentoimivista ratkaisuista. Hissiverkon topologia voidaan määritellä jo suunnitteluvaiheessa, eikä yleisiä verkonhallintaohjelmia enää asennusvaiheessa tarvita. Verkonhallintaohjelman sijasta Lon-verkon osoitteet ja verkkomuuttujien valitsimet asetetaan valmiiksi jo Neuron-piirin ohjelmakoodissa.

Tasapainoisen kokonaiskuvan luomiseksi valmistellulla työkurssilla esitellään myös standardit verkkomuuttujat omalla pienimuotoisella laitteistolla. Tällöin solmujen väliset loogiset kytkennät luodaan verkonhallintaohjelmistolla, mikä edustaa Lon-verkkojen tyypillisintä toteutustapaa.

7 TYÖKURSSIN HARJOITUSTÖIDEN SUUNNITTELU

Työssä toteutettua hissinohjausjärjestelmän simulointiympäristöä varten saatiin lahjoituksena oikean hissin ohjauselektroniikkaa. Laitteistokokonaisuuden määrittelyn yhteydessä jouduttiin muuttamaan alkuvaiheen suunnitelmaa työkurssin sisällöstä. Ensimmäisissä hahmotelmissa työkurssin harjoitukseksi kaavailtiin hissin perusohjauksen toteuttamista. Tällä olisi hallittu hissin ajologiikkaa ja kutsujen kirjausta. Laitetoimittaja totesi tämän liian laajamittaiseksi tehtäväksi hissin perusohjauksen ohjelmakoodin laajuudesta johtuen. Ohjelmoinnin kohteeksi valittiin painonappeja ja näyttöä ohjaavia kontrollereita, joilla annetaan hisseille kori- ja kohdekutsuja. Nämä muodostavat sopivia kokonaisuuksia, joista saadaan lisäksi havainnollisia näyttöjen ja nappien indikaattorivalojen ansiosta.

Kori- ja kohdekutsupaneelien Neuron-piirien tärkeimpänä tehtävänä on viestien kääntäminen kenttäväylän, nappien ja näytön välillä, kun painonapeilla annetaan kutsuja ympäristön simuloimille kahdelle hissikorille. Näitä tukemaan tarvitaan muuta ohjelmistoa ja laitteistoa, jotta ympäristö on elävä ja havainnollinen. Simulointiympäristöä on esitelty tarkemmin liitteessä 6.

Diplomityöhön sisältyi sovellusohjelmien suunnittelu ohjelmoinnin kohteiksi valituille Neuron-piireille sekä näiden sovellusohjelmien muokkaaminen työkurssin harjoitustöiksi. Tavoitteena on ollut mahdollisimman yksinkertaisten ja selkeiden sovellusohjelmien tuottaminen kohdepiireille. Harjoitustöiden edellyttämät eri vaikeusasteet on pyritty toteuttamaan siten, että tehtävänannon pakollinen osa toteuttaa sovelluksen perustoiminnallisuuden, jota voidaan vapaaehtoisesti täydentää lisäominaisuuksilla. Kehitystyön tukena ovat olleet laitetoimittajan tarjoamat verkkoliikenteen rajapinnan määrittelyt sisältävät koodipohjat sekä laitteiden ja alkuperäisten sovellusohjelmien toiminnalliset kuvaukset.

Simulointiympäristön laitteiston alustavan määrittelyn jälkeen suunniteltiin erilliset harjoitukset kori- ja kohdekutsupaneeleita varten. Nämä harjoitukset ovat liitteinä 2. ja 4. Lisäksi työkurssi suunniteltiin aloitettavaksi erillisellä vilkkuvaloharjoituksella, jonka sovellusohjelma ei vielä käytä verkkoliikenteen toimintoja. Se kaavailtiin suoritettavaksi koripaneelin Neuron-piirillä.

Echelonin toimittamaa Neuron C -ohjelmointiopasta (Echelon 2003b) luettaessa kävi ilmi, ettei laitetoimittajan tapa hyödyntää LonWorks-tekniikkaa ole tavanomainen. LonWorks-tekniikan tarjoamia mahdollisuuksia standardin verkkoliikenteen rajapinnan toteuttamiselle ei ole hyödynnetty. Työkurssin laaja-alaisuuden turvaamiseksi laitteistoon pyydettiin erillistä osaa, jossa standardeja verkkomuuttujia voitaisiin kokeilla varsinaista hissin toimintaa simuloivaa laitteistoa häiritsemättä. Tämä lisäys saatiinkin simulointiympäristön lopulliseen laitteistoon, ks. liite 6. kuvat 1. ja 2. Verkkoliikenteen standardin rajapinnan esittelevä harjoitustyö on liitteenä 5.

Työkurssin ensimmäinen harjoitus siirrettiin nyt suoritettavaksi simulointiympäristön erilliseen osaan, jossa se voidaan suorittaa mahdollisimman turvallisesti. Tämä vilkkuvaloharjoitus on liitteenä 1. Vaikka tässä harjoituksessa tuotettu sovellusohjelma ei vielä hyödynnä verkkotoimintoja, siirretään sovellusohjelma kohdepiiriin verkonhallintaohjelmalla, jonka käyttöä neuvotaan liitteessä 6.

Simulointiympäristön erilliseen osaan suunniteltiin vielä eksplisiittisiin sovellusviesteihin tutustuttava harjoitus. Tässä viestintätekniikassa ei käytetä verkkomuuttujien tarjoamaa automaattista viestinvälitystä, vaan sovellusohjelman kirjoittaja huolehtii viestin kirjoittamisesta ja lähettämisestä. Sovellusviesteihin on luontevaa tutustua ensin laitteiston erillisen osan eristetyssä ympäristössä liitteen 3.

harjoituksella. Sovellusviestien käyttötaitoa tarvitaan heti seuraavassa liitteen 4. kohdekutsuharjoituksessa.

Harjoitustöissä tuotetaan ohjelmakoodia simulointiympäristön asettamat rajoitukset ja ehdot huomioiden. Neuron-piirin asetukset sekä hissiverkossa käytettävät verkkomuuttujat annetaan kurssilaisille valmiina. Ne edustavat ympäröivän laitteiston rajoitteita, joihin ohjelmoijan tulee mukautua. Kahdessa viimeisessä harjoituksessa myös painonappeja ja LCD-näyttöä ohjaavat funktiot annetaan valmiina. Näin sovellusohjelma saa varmasti oikean syötteen ja näyttö tarjoaa palautteen oikein. Tällöin edellytykset sovelluksen toiminnan tarkkailuun ovat olemassa, jolloin voidaan keskittyä ohjelman varsinaisen toimintalogiikan hiomiseen. Tällä myös turvataan eritasoisten opiskelijoiden mahdollisuus päästä harjoitustöissä alkuun. Harjoitustöiden vaativin osuus tarjotaan monimutkaisemman toimintalogiikan muodossa, joka edellyttää esim. ajastimien monipuolista käyttöä. Näin harjoitustöiden painopiste siirtyy paikallisten I/O-toimintojen hallinnasta reaaliaikaisen järjestelmän ajoituksen ja toimintalogiikan suunnitteluun.

7.1 Harjoitustöiden tehtävänantoja testaava erikoistyö

Harjoitustöiden työmäärän arvioimiseksi sekä työohjeiden toimivuuden varmistamiseksi järjestettiin erikoistyö, jonka tekijä suoritti liitteiden 1. ja 5. harjoituksia. Erikoistyön tekijältä saadun palautteen mukaisesti harjoitustöiden tehtävänantoja kehitettiin selkeämmiksi. Yleinen informaatio annetaan edelleen kuvailevaa sanamuotoa käyttäen, mutta työn sisältöä määrittävät kohdat ovat nyt käskevässä muodossa ja mahdollisesti lihavoituna.

Lisäksi tehtävänantojen sisältämät viittaukset kurssin muuhun materiaaliin on nyt tehty mahdollisimman tarkasti. Ensimmäiseen harjoituksen tehtävänantoon lisättiin erityinen järjestyslista suoritettavista toimenpiteistä ja niissä tarvittavien tietojen sijainnista, koska suuri osa harjoituksessa tarvittavista tiedoista sijaitsee muissa dokumenteissa.

Erikoistyön perusteella harjoitustöiden työmäärä näytti aiemmin arvioitua suuremmalta, mutta tästä ei saatu täyttä varmuutta. Kurssia on tarkoitus suorittaa kahden hengen ryhmissä, jolloin ryhmän jäsenet kykenevät auttamaan toisiaan. Tällä on oletettavasti suuri merkitys kurssin harjoitustöissä, joissa on pakko edetä vaiheittain ongelma kerrallaan ratkaisten.

7.2 Sovellusohjelmien suunnittelussa esiintyneet ongelmat

Harjoitustöiden pohjina toimivia sovellusohjelmia kehitettäessä ongelmia aiheutti Neuron-piirin haavoittuvuus sovellusohjelman ohjelmointivirheille, joita ohjelmointityökalu ei havaitse ajoissa. Kerättyjen kokemusten perusteella laadittiin erityiset turvaohjeet, jotka ovat liitteen 6. lopussa.

Jotta työkurssin suorittajat hallitsevat turvaohjeiden sisällön, on kurssille tulemisen edellytykseksi kaavailtu esiselostustehtävien suorittamista. Toisin kuin esitentillä, esiselostustehtävillä voidaan aktiivisesti ohjata tutustumisen kohteena olevia asioita.

8 YHTEENVETO

Tässä diplomityössä määriteltiin opetuslaitteisto sulautettujen järjestelmien työkurssin käyttöön, suunniteltiin sovellusohjelmat laitteistolle ja muokattiin ne edelleen työkurssin harjoitustöiksi.

Opetuslaitteistossa tavoitteena ollut näyttävyys ja havainnollisuus on saavutettu näyttöjen ja merkkivalojen avulla kiitettävästi. Ne tarjoavat ohjelmoijalle palautetta kehitettävän sovelluksen toiminnasta ja simuloitujen hissien liiketiloista. Harjoitustöiden osalta tavoitteena olleet erilaisia vaikeusasteita sisältävät tehtävät ovat myös toteutuneet suunnitellusti. Sovelluksille on asetettu vähimmäisvaatimukset perustoiminnallisuuden toteuttamiseksi, ja ohjelmia voidaan haluttaessa kehittää eteenpäin ominaisuuksia lisäämällä

Opetusmateriaalin kehityksessä tavoitteena ollut laaja-alaisuus on toteutunut tyydyttävästi. Työkurssin opetuksellinen anti ei rajoitu hissinohjaukseen, sillä kurssilla keskitytään kokonaista hissinohjausjärjestelmää pienempiin yksityiskohtiin, kuten painonappien ja näytön ohjaukseen ja verkkoliikenteen rajapintoihin. Nämä taidot ovat hyvin yleishyödyllisiä, sillä vastaavia sovelluskohteita löytyy runsaasti.

Kenttäväylän välityksellä tapahtuvassa viestinnässä ei myöskään rajoituta tarkastelemaan pelkästään LonWorks-tekniikan erityispiirteitä. Verkkomuuttuja on korkean tason erikoispalvelu, jota ei ole tarjolla muissa teknologioissa. Sovellusohjelmassa eksplisiittisesti muodostettavat ja lähetettävät viestit edustavat laajemmassa käytössä olevaa menetelmää, joten niihin tutustuminen on myös pedagogisesti perusteltua yleistietojen kartuttamiseksi.

Työkurssin laitteiston monipuolisen kokoonpanon ansiosta kurssilla voidaan tutustua monipuolisesti LonWorks-tekniikan erilaisiin soveltamistapoihin. Hissin toimintaa simuloiva laitteiston pääosa edustaa räätälöityä valmistajakohtaista erityisratkaisua, joka poikkeaa eri valmistajien kesken yhteentoimivista ratkaisuista. Siinä ei ole hyödynnetty LonWorks-tekniikan tarjoamia mahdollisuuksia standardin verkko-liikenteen rajapinnan toteuttamiseksi. Laitteiston erillisen osan ansiosta kurssilla päästään kokeilemaan myös standardin rajapinnan luomista, jolloin kurssin luoma kokonaiskuva LonWorks-tekniikan ominaisuuksista kyetään pitämään tasapainoisena.

Standardin verkkoliikenteen rajapinnan komponentit generoidaan automaattisesti NodeBuilder-ohjelmiston sisältämällä CodeWizard-ohjelmalla. Koska ohjelmointityö kehittyy tulevaisuudessa yhä enemmän automaattisia työkaluja hyödyntävään suuntaan, on niihin tutustuminen perusteltua myös työkurssilla.

Työkurssin laitteiston erillinen osa osoittautui erinomaiseksi lisäksi varsinaiseen hissin simulointiympäristöön. Standardin rajapinnan harjoituksen lisäksi siinä voidaan tehdä ensimmäinen harjoitus turvallisesti varsinaisen hissiverkon toimintaa vaarantamatta. Myös sovellusviesteihin tutustuttava harjoitus on selkeintä tehdä omassa muista häiriötekijöistä eristetyssä ympäristössä, jossa kuitenkin pystyy seuraamaan oman viestin lähettämistä ja vastaanottamista.

Ohjelmointiympäristön haavoittuvuus ohjelmointivirheille tulee vaatimaan työtä laitteiston ylläpitämiseksi, mutta muuten simulointiympäristö täyttää sille asetetut tavoitteet erinomaisesti.

Työkurssin harjoitustöiden suorittamisen vaatimaa työmäärää on ollut vaikea arvioida harjoitustöitä kehitettäessä. Tästä saadaan palautetta myöhemmin työkurssin järjestämisen yhteydessä, jolloin voidaan tehdä korjauksia kurssista myönnettävään

opintopisteiden määrään. Myös harjoitustöissä kehitettävien sovellusten pohjaksi annettavia mallitiedostoja voidaan kehittää eteenpäin kertyvän kokemuksen perusteella. Mallitiedoston sisältämän ohjelmakoodin ja vihjeiden määrällä voidaan tehokkaasti säädellä harjoitustyön vaativuutta.

Kehitettyä simulointiympäristöä on tarkoitus käyttää opetukseen seuraavat viisi vuotta. Tänä aikana simulointilaitteiston tekniikka ehtii muuttua osin vanhan-aikaiseksi. Jo nyt LonWorks-tekniikkaa pidetään hieman vanhentuneena teknologiana, ja kontrollereiden ohjelmoinnissa siirrytään vähitellen oliopohjaisiin korkeamman tason ohjelmointikieliin. Toisaalta varsinkin hissiteollisuudessa luotettavuus on keskeinen lähtökohta, jolloin on perusteltua turvautua koeteltuihin ratkaisuihin. Lisäksi hisseihin asennettujen ohjausjärjestelmien käyttöikä on yli 20 vuotta, mikä asettaa vaatimuksia komponenttien saatavuudelle vuosiksi eteenpäin.

Sulautettujen järjestelmien työkurssi tulee tarpeeseen ja sen tarjoamat taidot vastaavat hyvin työmarkkinoilla vallitsevaan kysyntään.

LÄHDELUETTELO

ANSI 1989. American National Standard X3.159-1989, Programming Language C, D.F. Prosser, American National Standards Institute, 1989.

Anyong, LonTalk Protocol Specification, 2000, 17 s. palgong.kyungpook.ac.kr/~anyong/home/seminar/lonwork/LonProtocol1.ppt (viitattu 15.12.05)

Apte Consulting Group, Network Detective: Reverse-engineering a network, koulutusmateriaalia, 1998, 51 s.

Atmel, AT29C256, Rev. 0046P-flash-10/04, flash-muistin datalehti, 2004, 17 s.

Belimo 2005, LonWorks Glossary, Belimo Automation AG, www.belimo.ch/pdf/e/Glossar_Lon_e.pdf (viitattu 26.4.06)

Cypress, CY7C53150 Neuron Chip External Memory Interface, 2000, 16 s.

Dietrich, D., Loy, D., Scheinzer, H., Open Control Networks, LonWorks/EIA 709 Technology, Kluwer Academic Publishers, 2001, 367 s.

Echelon 2006, Self-Installation Policy and Guidelines, echelon.com/support/service/policies/selfinstall.htm (viitattu 21.2.06)

Echelon, FT 3120 / FT 3150 Smart Transceiver Data Book, 005-0139-01B, 2004, 211 s.

Echelon, Host Application Programmer's Guide, 078-0016-01B, 1993, 126 s.

Echelon, LonWorks Custom Node Development, Engineering Bulletin, 1995a, 16 s.

Echelon, LonWorks Technology, Intelligent Distributed Control Training Course, v3.2, 1995b.

Echelon, Microprocessor Interface Program (MIP) User's Guide, 078-0017-01C, 1995c.

Echelon, Neuron C Reference Guide, 078-0140-02E, 2003a, 364 s.

Echelon, Neuron C Programmer's Guide, 078-0002-02G, 2003b, 328 s.

Echelon, NodeBuilder User's Guide, 078-0141-01E, 2003c, 378 s.

Graham 2006, LonWorks History and Technology Overview, Graham Controls Consulting, Inc., www.grahamcontrols.com/LWOverview.htm (viitattu 26.4.2006)

Harbinson S. P., Steele G. L., C: A Reference Manual, Prentice-Hall, Inc., 1991.

ISO 1981, International Organization for Standardization, Open Systems Interconnect - Basic Reference Model, IS 7498, 1981.

LonMark 2006, LONMARK Functional Profiles, www.lonmark.org/products/fprofile.htm, (viitattu 26.4.2006)

Miller L. H., Quilici A. E., The Joy of C, John Wiley & Sons, Inc, 1997, 788 s.

Motorola, Installation of Neuron Chip -Based Products, AN1276, 1997a, 16 s.

Motorola, LonWorks Technology Device Data, Rev. 3.1, 1997b, 322 s.

Motorola, MC683xx to Neuron Chip Parallel I/O Interface, AN1247, 1997c, 21 s.

Piikkilä Veijo, keskustelut 8.12.2005.

Piikkilä Veijo, LonWorks-tekniikan perusteet, Tammertekniikka, 2004.

Piikkilä Veijo, Rakennusautomaation kenttäväylän liittäminen internetiin, Diplomityö, Tampereen teknillinen korkeakoulu, Automaatiotekniikan osasto, 2002, 125 s.

Toshiba 2006, TAEC Neuron Chip Databook - TMPN 3120/3150, www.toshiba.com/taec/components/Generic/DB_neuron.jsp (viitattu 24.4.2006)

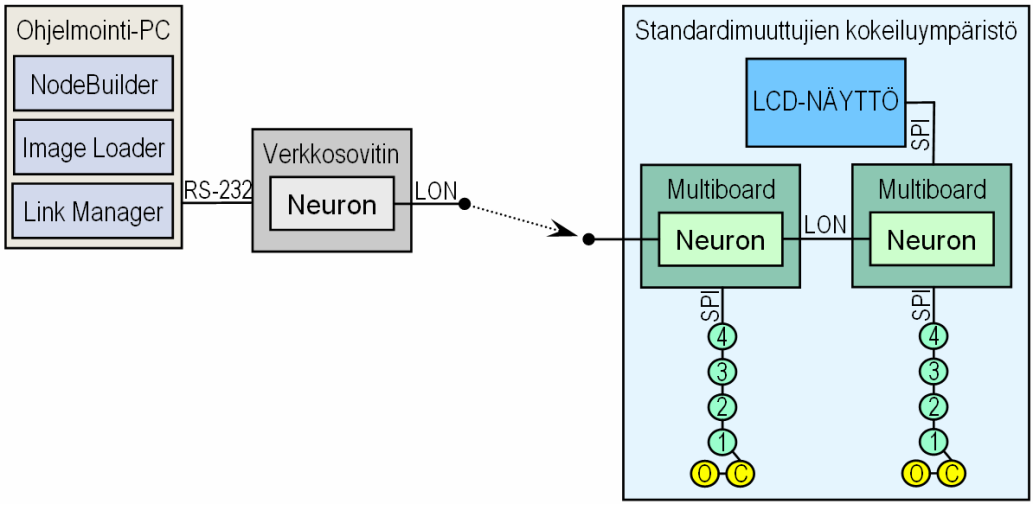
Toshiba, TMPN3120FE5M Data Sheet, 2001, 11 s.

S-81.3210 Sulautettujen järjestelmien työkurssi (5 op)

Liite 1. HARJOITUS: Vilkkuvalot

Työkurssin ensimmäisen harjoitustyön tavoitteena on tuottaa ajokelpoinen sovellusohjelma Neuron-piirille. Sovellusohjelman toimintaa ohjataan Neuron-piirin SPI-väylään kytketyillä painonapeilla ja niiden indikaattorivalot antavat palautetta ohjelman toiminnasta.

Harjoitustyössä käytetään simulointiympäristön erillistä osaa, joka sisältää kaksi Neuron-piiriä kuvan 1. mukaisesti. Työssä keskitytään vain yhden Neuron-piirin toimintaan, joten kehitettävä sovellusohjelma ei vielä käytä Lon-verkkoa. Kehitetty ohjelmakoodi ladataan vasemmanpuoleisen Multiboard-piirilevyn Neuroniin.



Kuva 1. Harjoitustyössä käytettävä laitteisto.

Sovellusohjelman rakenne perustuu tapahtumapohjaiseen ajoitukseen, jossa when-lauseita ohjataan ajastimilla. Tutustu ohjelmointikieleen seuraavia dokumentteja käyttäen:

- ”Neuron C -ohjelmointi”, luku 4. työkurssin valmistelleesta diplomityöstä. Erityisesti tapahtumapohjainen ajoitus, ajastimet sekä I/O-toiminnot.
- ”Neuron C Programmers Guide”, yleinen Neuron C -ohjelmointiopas. Lue ensimmäistä lukua sivulle 8. asti sekä toista lukua sivulle 16. saakka.
- ”Neuron C Reference Guide”, hakuteoksena käytettäväksi. Dokumentti sisältää ohjeet Neuron-piirin varusohjelmiston funktioiden käyttöön.

Tässä harjoitustyössä tarvittavien ohjelmointi- ja verkonhallintatyökalujen käyttö neuvotaan ”Simulointiympäristön käyttöohjeessa”. Se on työkurssin valmistelleen diplomityön 6. liite.

Harjoitustyössä tuotettavaa ohjelmakoodia varten annetaan pohjatiedosto, jota käytetään ohjelmakoodin kirjoituksen pohjana. Pohja sisältää kääntäjän vaatimat määrittelyt sekä mallin I/O-funktion käytöstä.

SUORITUS: Ryhmä varaa työskentelyajat simulointilaitteiston luona olevasta varauslistasta. Työaikoja saa varata korkeintaan kolmeksi tunniksi kerrallaan. Työtä suositellaan tehtäväksi määräaikoihin nähden etupainotteisesti, jotta vältytään ruuhkilta laitteiston käytössä.

Tehty harjoitus hyväksytetään assistentilla erikseen määrättyinä esittelyaikoina. Viimeisen onnistuneesti toteutetun vaiheen esittely riittää. Esittelyajat ja harjoituskohtaiset aikataululliset takarajat käyvät ilmi varauslistasta. Onnistuneen esittelyn jälkeen harjoitustyön viimeisen hyväksytyn vaiheen lähdekoodi lähetetään vielä assistentille sähköpostin liitetiedostona. Lähettämistä varten tiedosto nimetään muotoon ”A_B_C.nc”, jossa A = ryhmän numero, B = harjoitustyön numero ja C = vaiheen numero. Sähköposti nimetään muotoon ”Ryhmä A, harjoitustyö B” ja varustetaan jäsenten nimillä ja opiskelijanumeroilla.

ARVOSTELU: Kurssin harjoitustyöt koostuvat pakollisista sekä vapaaehtoisista osista. Seuraavaan harjoitustyöhön ei saa siirtyä, ellei edellisen harjoitustyön pakollista osaa ole suoritettu.

Kurssiarvosana määräytyy vapaaehtoisista tehtävistä kerättyjen pisteiden perusteella taulukon 1. mukaisesti. Suorittamalla kaikista kurssin harjoitustöistä pelkän pakollisen osuuden saa arvosanaksi ykkösen. Viitosen saadakseen tulee kerätä vähintään 80 % suorituspisteistä. Pisteet ilmoitetaan prosentteina, sillä jos jonkin kurssin harjoitustyön suoritus estyy teknisistä syistä, lasketaan suoritettujen pisteiden osuus käytännössä tarjolla olleista maksimipisteistä. Vapaaehtoisten tehtävien lukumäärä vaihtelee eri harjoituksissa.

Taulukko 1. Kurssiarvosanan määräytyminen.

Arvosana	1	2	3	4	5
Pisteet	<20 %	≥20 %	≥40 %	≥60 %	≥80 %

Ensimmäisessä harjoitustyössä on kolme vaihetta. Niistä kaksi ensimmäistä ovat pakollisia, ja viimeisestä saa suorituspisteitä. Viimeisestä vaiheesta on mahdollista saada kaksi pistettä, mikäli tehtävänanto toteutuu kaikilta osin.

Ohjelmointi- ja verkonhallintatyökalujen käyttöönotto ”Simulointiympäristön käyttöohjeen” mukaisesti etenee seuraavasti:

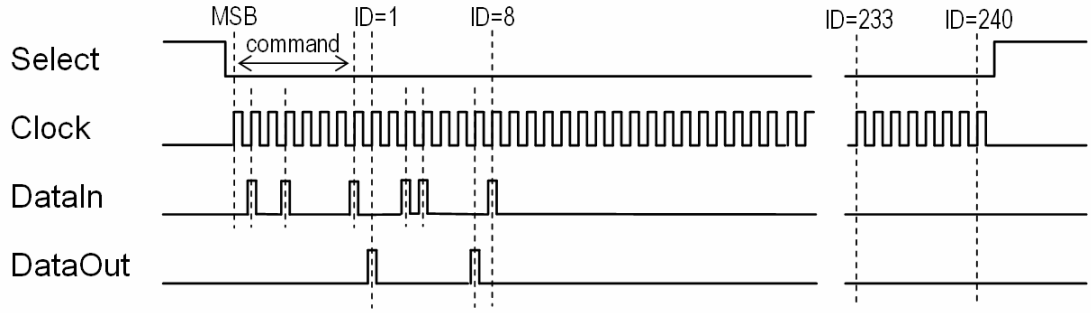
1. Käynnistä NodeBuilder ja luo siihen oma projektisi (ohjeet luvussa 2.1).
2. Kopioi annettu pohjatiedosto projektihakemistoosi (ohjeet luvussa 2.1).
3. Luo oma sovellusohjelmasi pohjatiedoston päälle NodeBuilderin editorilla. Ohjeet sovellusohjelman tekoon löytyvät tästä dokumentista.
4. Käännä ohjelmasi binääritiedostoksi NodeBuilderissa (ohjeet luvussa 2.1).
5. Käynnistä laitteisto ja SLTALink Manager (ohjeet luvuissa 2.2 ja 2.3).
6. Siirrä kääntämäsi binääri Neuron-piiriin Image Loaderilla (ohjeet luvussa 2.4).

1. VAIHE (pakollinen): Luodaan sovellusohjelma, joka kytkee SPI-väylän painonappien indikaattorivaloja päälle. Tavoitteena on saada valo kiertämään ympyrää siten, että yksi valo palaa kerrallaan ja siirtyy järjestyksessä seuraavaan nappiin sekunnin välein.

Ohjelmakoodia voidaan kehittää vaiheittain, jolloin ensimmäisessä versiossa yksi tai useampi napin valo palaa jatkuvasti. Tämä on ensimmäinen askel nappien SPI-väylän käyttöön. Seuraavassa versiossa yhden napin valo vilkkuu, mikä harjaannuttaa ajastimien käyttöön. Viimeisessä versiossa napin valo siirtyy seuraavaan nappiin ja kulkee ympyrää. Myös OPEN- ja CLOSE-nappien valot tulee sisällyttää ohjelmaan, jolloin ympyrä on: …OPEN, CLOSE, 1, 2, 3, 4, OPEN…

SPI-väylä (Serial Peripheral Interface) on nelijohtiminen synkroninen sarjamuotoinen tiedonsiirtoväylä kontrollerin ja oheislaitteen välillä. Siirron toinen osapuoli toimii isäntänä (master) ja antaa kellosignaalin. Data siirtyy molempiin suuntiin yhtä aikaa.

Työkurssin simulointiympäristössä napeille lähetettävä SPI-viesti on aina 31 tavun mittainen. Ensimmäinen tavu sisältää SPI-väylän ohjauskäskyn. Harjoitustöissä käytetään aina käskyä 0x51, jolla tunnistetaan nappien asennot ja hallitaan nappien valoja. Loput 30 tavua sisältävät 240 bittiä, joista jokainen ilmaisee yhden napin tai sen valon tilaa. Jokaisella napilla on oma tunnusnumero sen mukaisesti, monettako ohjauskäskyn jälkeistä bittiä ne lukevat/kirjoittavat. SPI-väylässä ensimmäinen komentotavun jälkeinen bitti vastaa ID-numeroa 1, toinen ID-numeroa 2 jne.



Kuva 2. Tiedonsiirto SPI-nappiväylässä. MSB = tavun merkitsevin bitti.

Kuvassa 2. SPI-väylän johtimeen ”DataIn” on kirjoitettu komentotavuksi 0x51. Valot palavat napeissa, joiden tunnusnumerot ovat 3, 4 ja 8. Painettuina ovat napit, joiden tunnusnumerot ovat 1 ja 7.

Simulointiympäristön painonappien tunnusnumerot vastaavat kerrosnumeroita, eli 1. kerroksen kutsunapin tunnus on 1 jne. Oven avausnapin tunnus on 113 ja sulkunapin 114.

SPI-väylään kirjoittaminen ja lukeminen tapahtuvat samalla kerralla, kun kutsutaan funktiota ”io_out( )”. Funktio lukee nappivalojen arvot sille annetusta datavektorista, jonka jälkeen se kirjoittaa nappien asentotiedot samaan vektoriin. Datavektorin jokainen tavu ilmoittaa kahdeksan napin tai nappivalon tilan. On huomioitava, että Neuron-piirissä binäärilukuesitysten merkitsevin bitti sijaitsee vasemmalla. Mikäli indikaattorivalojen päälle kytkeminen tuottaa alkuvaiheessa ongelmia, voidaan kaikkien tavun bittien arvoksi asettaa 1 antamalla tavun arvoksi 0xFF. Datavektorin ensimmäinen tavu sisältää aina SPI-käskyn, joka on painonapeille aina 0x51.

Muista I/O-porteista poiketen SPI-väylä ei ilmaise mahdollista sisääntuloarvon muutosta, eli liipaisufunktioita ”io_changes( )” tai ”io_update_occurs( )” ei voida käyttää. Sen sijaan SPI-väylää tulee lukea säännöllisesti 50 millisekunnin välein.

2. VAIHE (pakollinen): Lisätään edellisen vaiheen ohjelmaan nappien painalluksilla suoritettu ohjaus. Painettaessa nappia 1 valo alkaa kiertää ympyrää suurempia nappien numeroarvoja kohti. Vastaavasti painettaessa nappia 2 kiertosuunta on kohti pienempiä numeroarvoja. Kiertosuuntaa voi vaihtaa koska hyvänsä painamalla nappia 1 tai 2.

3. VAIHE (vapaaehtoinen, 2 p): Lisätään ohjelmaan napeilla aktivoitavia toimintoja.

Nappi 1: valo juoksee ympyrää ylöspäin
Nappi 2: valo juoksee ympyrää alaspäin
Nappi 3: liikenopeus kasvaa
Nappi 4: liikenopeus hidastuu
OPEN: valo juoksee edestakaisin, myös napeissa OPEN ja CLOSE
…OPEN, CLOSE, 1, 2, 3, 4, 3, 2, 1, CLOSE, OPEN, CLOSE, 1, 2…
CLOSE: nappien valot sammuvat

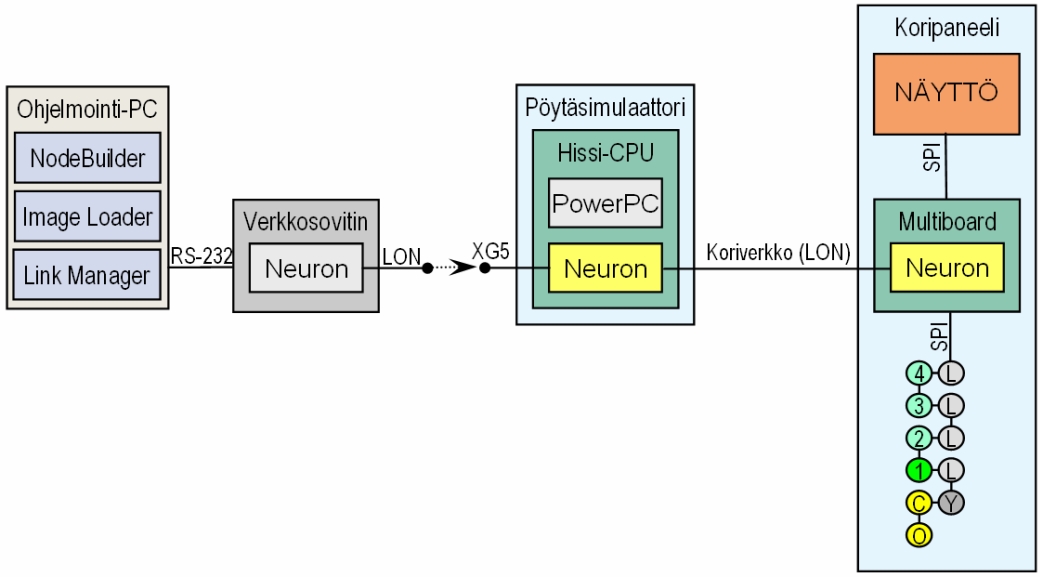
Toimintatilaa on kyettävä muuttamaan missä järjestyksessä tahansa. Uuden toimintatilan ei tarvitse tulla voimaan välittömästi, vaan se saa tulla voimaan vasta valon saavuttua napille 4 tai CLOSE.

Muutettaessa liikenopeutta ei valon kulku saa katketa, vaan sen tulee jatkaa eteenpäin vallitsevan toimintatilan mukaisesti ainoastaan muuttuneella liikenopeudella.

Liite 2. HARJOITUS: Korikutsut

Työkurssin toisen harjoitustyön tavoitteena on tuottaa Neuron C -ohjelmakoodi hissin korikutsujen antamiseen. Kehitettävä sovellusohjelma ladataan koripaneelin Neuron-piiriin, jossa se kääntää viestit Lon-koriverkon ja Neuron-piirin SPI-väylän välillä. Koripaneelin kutsunapeilla annettu palvelupyyntö välitetään hissi-CPU:lle, jonka vastaus esitetään kutsunappien indikaattorivaloilla ja koripaneelin näytöllä.

Kurssin käytössä oleva hissin toimintoja simuloiva laitteisto on esitelty kokonaisuudessaan ”Simulointiympäristön käyttöohjeessa”. Tässä harjoituksessa käytetään laitteiston oikeanpuoleista pöytäsimulaattoria ja koripaneelia, jotka esitetään kuvassa 1. Tätä laitteiston osaa voidaan käyttää itsenäisesti muusta laitteistosta riippumatta, jolloin käyttöjännite kytketään vain oikean puoleiseen pöytäsimulaattoriin sen päällä olevasta katkaisijasta. Valmis korikutsuohjelma voidaan testata koko laitteistoa käyttäen, mutta sovelluksen kehitystyön aikana käytetään vain laitteiston kuvan 1. mukaista osaa.



Kuva 1. Harjoitustyössä käytettävä laitteisto.

Sovellusohjelmassa käytetään verkkomuuttujia Lon-verkon solmujen välisessä viestinnässä. Sovellusohjelma asettaa Neuron-piirin EEPROM-muistiin solmun oman verkko-osoitteen, viestien kohteena olevan solmun osoitteen sekä verkkomuuttujien valitsimet. Tutustu ohjelmointikieleen seuraavia dokumentteja käyttäen:

- ”Neuron C Programmers Guide”, yleinen Neuron C -ohjelmointiopas. Lue kolmatta lukua sivulle 14. saakka.
- ”Verkkoliikenteen asetustaulukot”, luku 6. työkurssin valmistelleesta diplomityöstä. Erityisesti ”Verkkoaluetaulukko” sekä ”Solmulähetys”
- ”Neuron C Reference Guide”, hakuteoksena käytettäväksi. Dokumentti sisältää ohjeet Neuron-piirin varusohjelmiston funktioiden käyttöön.

Harjoitustyössä tuotettavaa ohjelmakoodia varten annetaan pohjatiedosto, jota käytetään ohjelmakoodin kirjoituksen pohjana. Pohja sisältää verkkomuuttujien ja I/O-objektien määrittelyt sekä mallin I/O-funktion käyttöön ja verkkoasetusten tekemiseen.

Käytä edellisessä harjoituksessa luomaasi NodeBuilder-projektia. Kopioi projektisi lähdekoodihakemistoon tiedosto “pohja_2.nc” ohjelmointi-PC:n hakemistosta ”D:\Mallit” ja nimeä se muotoon ”group_x.nc” (x = ryhmänne numero) alkuperäisen projektitiedoston korvaamiseksi. Ohjelmointi- ja verkonhallintatyökalujen käyttö tapahtuu muuten edellisen harjoituksen mukaisesti.

SUORITUS: Ryhmä varaa työskentelyajat simulointilaitteiston luona olevasta varauslistasta. Työaikoja saa varata korkeintaan kolmeksi tunniksi kerrallaan. Työtä suositellaan tehtäväksi määräaikoihin nähden etupainotteisesti, jotta vältytään ruuhkilta laitteiston käytössä.

Tehty harjoitus hyväksytetään assistentilla erikseen määrättyinä esittelyaikoina. Viimeisen onnistuneesti toteutetun vaiheen esittely riittää. Esittelyajat ja harjoituskohtaiset aikataululliset takarajat käyvät ilmi varauslistasta. Tämän jälkeen hyväksytysti esitellyn vaiheen lähdekoodi lähetetään vielä assistentille sähköpostin liitetiedostona. Lähettämistä varten tiedosto nimetään muotoon ”A_B_C.nc”, jossa A = ryhmän numero, B = harjoitustyön numero ja C = vaiheen numero. Sähköposti nimetään muotoon ”Ryhmä A, harjoitustyö B” ja varustetaan jäsenten nimillä ja opiskelijanumeroilla.

ARVOSTELU: Kurssin harjoitustyöt koostuvat pakollisista sekä vapaaehtoisista osista. Seuraavaan harjoitustyöhön ei saa siirtyä, ellei edellisen harjoitustyön pakollista osaa ole suoritettu.

Kurssiarvosana määräytyy vapaaehtoisista tehtävistä kerättyjen pisteiden perusteella taulukon 1. mukaisesti. Suorittamalla kaikista kurssin harjoitustöistä pelkän pakollisen osuuden saa arvosanaksi ykkösen. Viitosen saadakseen tulee kerätä vähintään 80 % suorituspisteistä. Pisteet ilmoitetaan prosentteina, sillä jos jonkin kurssin harjoitustyön suoritus estyy teknisistä syistä, lasketaan suoritettujen pisteiden osuus käytännössä tarjolla olleista maksimipisteistä. Vapaaehtoisten tehtävien lukumäärä vaihtelee eri harjoituksissa.

Taulukko 1. Kurssiarvosanan määräytyminen.

Arvosana	1	2	3	4	5
Pisteet	<20 %	≥20 %	≥40 %	≥60 %	≥80 %

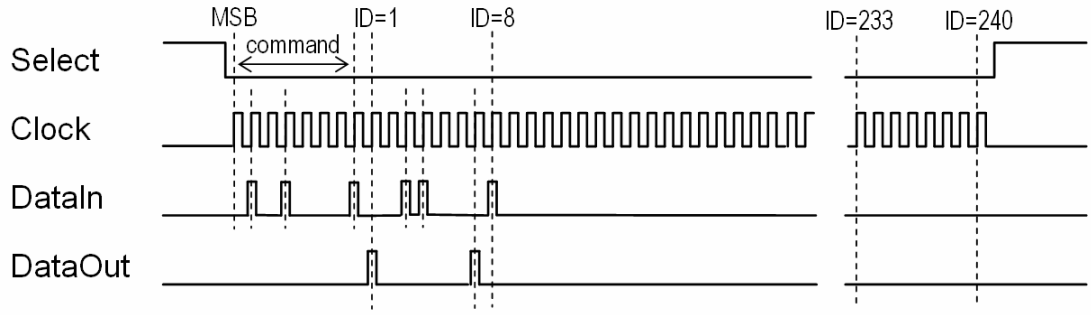
Toisessa harjoitustyössä on kolme vaihetta. Niistä kaksi ensimmäistä ovat pakollisia ja viimeisestä saa suorituspisteitä. Kolmannesta vaiheesta on mahdollista saada jopa kolme pistettä, mikäli tehtävänanto toteutuu kaikilta osin.

Tässä harjoituksessa tuotetun sovellusohjelman toimintalogiikkaa hiotaan vaiheittain paremmaksi. Työ koostuu kolmesta peräkkäisestä vaiheesta, joissa ensimmäisen toimivan version ominaisuuksia täydennetään niin pitkälle kuin intoa ja taitoa riittää.

1. VAIHE (pakollinen): Tehdään korikutsuohjelman perusversio, jossa napeilla annetut kutsut siirretään SPI-väylästä verkkomuuttujaan. Hissi-CPU:n palauttamat indikaattorivalokäskyt siirretään verkkomuuttujasta SPI-väylään. Lisäksi hissi-CPU ohjaa koripaneelin näyttöä, joten tämä ohjausdata siirretään verkkomuuttujasta näytölle myös SPI-väylää käyttäen.

Koripaneelin tärkeimpiä paikallisia I/O-toimintoja ohjataan Neuron-piirin jaetun SPI-väylän kautta. Jokaiselle SPI-väylää käyttävälle toiminnolle on oma Neuron-piirin I/O-pinni, jolla väylä varataan kyseisen toiminnon käyttöön. Pohjatiedoston sisältämien I/O-määrittelyjen ansiosta nämä valintasignaalit kytkeytyvät automaattisesti I/O-funktiota käytettäessä.

Painonappeja ja niiden indikaattorivaloja ohjataan edellisessä harjoituksessa opitulla tavalla. Nappeja ohjattaessa viestin pituus on aina 31 tavun mittainen. Ensimmäinen tavu sisältää käskyn 0x51 ja loput 30 tavua sisältävät 240 bittiä, joista jokainen ilmaisee yhden napin tai sen valon tilaa. SPI-väylässä ensimmäinen komentotavun jälkeinen bitti vastaa ID-numeroa 1, toinen ID-numeroa 2 jne.



Kuva 2. Tiedonsiirto SPI-nappiväylässä. MSB = tavun merkitsevin bitti.

Kuvassa 2. SPI-väylän johtimeen ”DataIn” on kirjoitettu komentotavuksi 0x51. Valot palavat napeissa, joiden tunnusnumerot ovat 3, 4 ja 8. Painettuina ovat napit, joiden tunnusnumerot ovat 1 ja 7.

Nyt ohjattavana ovat painonappien lisäksi kerroslukitukset. Lukittu kerros vastaa jatkuvasti pohjassa olevaa kutsunappia, eli saapuvassa SPI-viestissä lukituksen tunnusta vastaava bitti on 1. Nappiväylän kytkimillä on seuraavat tunnusnumerot:

1. kerros = 1
2. kerros = 2
3. kerros = 3
4. kerros = 4
Avaa = 113
Sulje = 114
1. lukitus = 129
2. lukitus = 130
3. lukitus = 131
4. lukitus = 132
Yksityiskäyttö = 219
Hälytysnappia ei ole kytketty SPI-väylään.

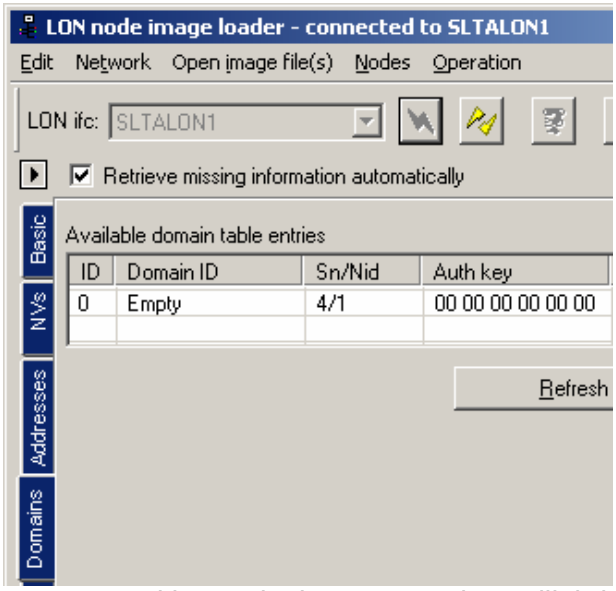
Verkkomuuttujissa kytkinten ja niiden valojen tilat esitetään myös 30-tavuisissa vektoreissa. On huomioitava, että verkkomuuttujissa päällä olevien bittien tulee antaa tavukohtaisesti oikea looginen numeroarvo. Jokainen tavu ilmoittaa kahdeksan kytkimen tai valon tilan. Kunkin tavun pienimmän kytkimen ollessa päällä on arvoksi tultava 1, toiseksi pienimmällä 2, kolmanneksi pienimmällä 4, neljänneksi pienimmällä 8 ja esim. kahden pienimmän napin samanaikaisella painalluksella 3. Neuron-piirissä binäärilukuesitysten merkitsevin bitti sijaitsee vasemmalla.

Koripaneelin Neuron-sovellus vastaanottaa näyttöä ohjaavaa dataa hissi-CPU:lta sisääntuloverkkomuuttujaansa, josta se ohjaa datan eteenpäin näytölle. Näyttöä ohjaava SPI-väylän viesti on aina kahdeksan tavun pituinen ja sen ensimmäinen tavu sisältää aina käskyn 0x81.

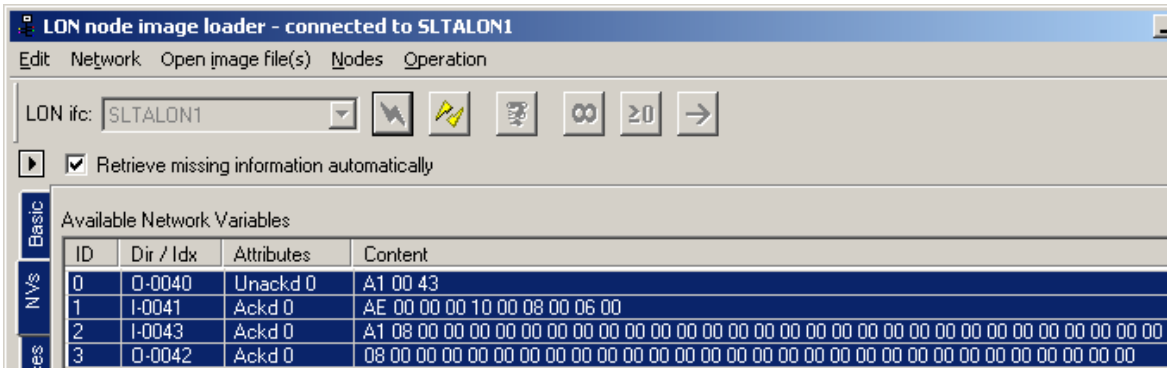
Näytön ohjausdataa vastaanottavan verkkomuuttujan kentän ”u8OldSpi” sisältämät kaksi tavua tulee siirtää SPI-viestin viidenteen ja kahdeksanteen tavuun. Kentän ”u8Pos” sisältämät kolme tavua tulee siirtää SPI-viestin toiseen, kuudenteen ja seitsemänteen tavuun vastaavassa järjestyksessä. Sovellusohjelma päivittää näyttöä vain tarvittaessa, eli vastaanottaessaan uutta ohjausdataa verkkomuuttujaansa.

Annetussa pohjatiedostossa on koripaneelin Neuronin omaksi verkko-osoitteeksi asetettu virheellisesti 0.2.3. (pääverkko.aliverkko.solmu). Kuva 3. esittää näkymän verkonhallintaohjelmaan tilanteessa, jossa verkko-osoite on oikea. Tee tarvittavat korjaukset ohjelmakoodiisi, jotta solmusi osaa vastaanottaa hissi-CPU:n lähettämät viestit. Tee korjaukset kuvan 3. perusteella, älä vielä lataa sovellustasi Neuron-piiriin. Pääverkon tunnuksessa merkintä ”empty” tarkoittaa, että tunnuksen pituudeksi on asetettu 0. Tällöin myös pääverkon tunnukseksi tulkitaan 0.

Koripaneelin Neuronin hissi-CPU:lle lähettämien viestien kohdeosoitteeksi on myös virheellisesti määritelty 0.2.3 (pääverkko.aliverkko.solmu). Etsi hissi-CPU verkonhallintaohjelmaan automaattista solmujen etsintää käyttäen. Hissi-CPU:n tunnus on ”CPU20101”. Tutki verkonhallintaohjelmalla hissi-CPU:lle määritelty looginen osoite ja määrittele tämä koripaneelista lähetettyjen viestien kohdeosoitteeksi ohjelmakoodiisi. Hissi-CPU:n sisältämästä kahdesta verkkoaluetaulukon merkinnästä valitaan ensimmäinen.



Kuva 3. Verkko-osoite löytyy Domains-välilehdestä. Sn = subnet, Nid = node ID.



Kuva 4. Verkkomuuttujien arvoja Lon Node Image Loader -ohjelmassa.

Kuvassa 4. on esitetty, miten korikutsun anto näkyy verkkomuuttujien arvoissa. Näkymä on Lon Node Image Loader -ohjelmasta, jolla ladataan NodeBuilderissa käännetty Neuron C -koodi Neuron-piiriin. Ohjelmaa voidaan kätevästi käyttää myös verkkomuuttujien arvojen seurantaan.

Kuvasta voidaan lukea verkkomuuttujien suunnat (Dir), valintatunnukset (Idx), palvelumuodot yhteydenpidossa (Attributes) sekä verkkomuuttujien arvot. Verkkomuuttujat näkyvät siinä järjestyksessä kuin ne on Neuron C -koodissa luotu. Valintatunnuksia käytetään muuttujien tunnistamiseen eri laitteiden välillä. Yhteydenpidon oletuspalvelumuotona toimii kuittaus (Ackd), jolloin verkkomuuttujaviestin vastaanottaja lähettää vahvistusviestin. Verkkomuuttuja voidaan myös asettaa etusijalle (Ackd P - priority), jolloin sen viestit saavat etusijan tiedonsiirtokanavan varauksessa.

Kuvassa kolmantena olevan hissi-CPU:lta indikaattorivalokäskyjä välittävän sisääntulomuuttujan arvoksi voidaan lukea 0xA1, 0x08 ja loput ovat nollia. Ensimmäisessä tavussa (A1) hissi-CPU ilmoittaa haluavansa saman tavun takaisin kuittauksena. Toisessa tavussa (08) hissi-CPU käskee kytkemään päälle nappivalon, jonka tunnus on neljä (koska heksadesimaaliluvun 0x08 binääriesityksessä on neljäs bitti päällä).

Kuvan neljäs muuttuja välittää korikutsut Neuron-piiristä hissi-CPU:lle. Sen arvosta (08) voidaan päätellä, että hissiä tilataan neljänteen kerrokseen.

Lon Node Image Loader -ohjelma ei näytä verkkomuuttujien arvojen muutosta jatkuvasti, vaan on painettava ohjelman Refresh-nappia. Tällöinkin arvojen lukemisessa on noin puolen sekunnin viive, mikä tulee kompensoida painamalla Refresh-nappia hieman ennen haluttua lukuhetkeä.

Hissi-CPU ei lähetä käskyjä oven avaus- ja sulkunappien valoille, vaan näitä valoja on tarkoitus ohjata Neuron-piirissä paikallisesti. Tämä toteutetaan työn seuraavassa vaiheessa.

Hyödyllisiä funktioita ja operaattoreita:

reverse( )	kääntää 8-bittisen luvun bitit toisin päin
& | ^	bittikohtaiset AND, OR ja XOR

Neuron-piirin varusohjelmiston funktioiden käyttöä opastetaan hakuteoksessa ”Neuron C Reference Guide”.

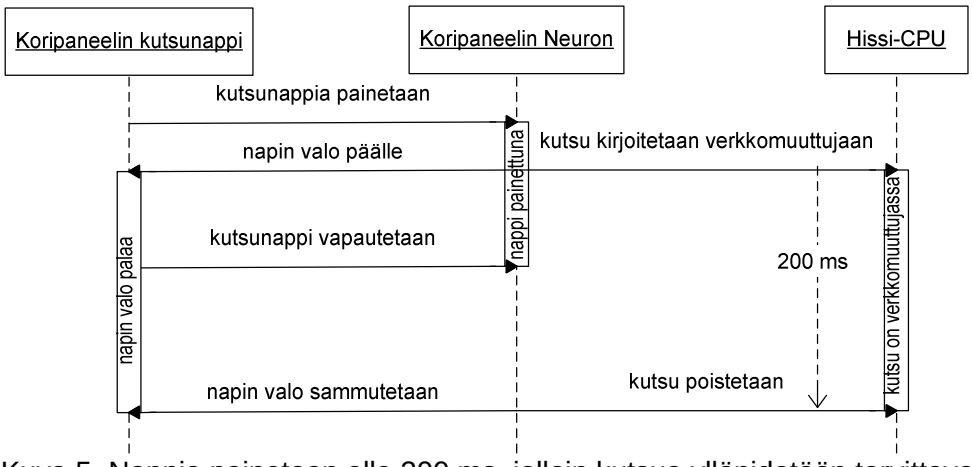
2. VAIHE (pakollinen): Toimivaan korikutsuohjelman perusversioon lisätään nk. direct-I/O -toiminto, jossa nappivalo syttyy välittömästi nappia painettaessa. Tällöin nappivaloja ohjataan hissi-CPU:n indikaattorivalokäskyjen lisäksi nappien asennon perusteella. Eli kaikkien pohjassa olevien nappien valot palavat ja lisäksi niiden kerrosnappien, joiden painalluksesta hissi-CPU on vastaanottanut korikutsun.

Nappien indikaattorivalot tulee kytkeä päälle korikutsut lähettävän verkkomuuttujan arvojen perusteella. Kun napin asento luetaan verkkomuuttujasta sen valon kytkemiseksi, voidaan verkkomuuttujaan kirjoitetun korikutsun päällä oloa tarkkailla indikaattorivalosta. Näin korikutsun poiskytkeytymisen viivettä voidaan havainnoida harjoitustyön seuraavassa vaiheessa.

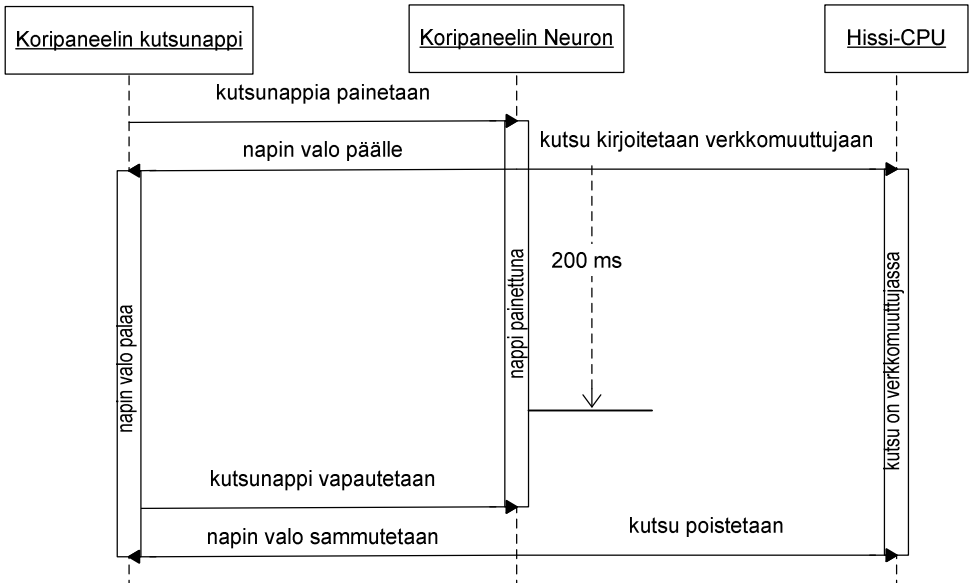
Kuten harjoituksen ensimmäisessä vaiheessa havaittiin, eivät oven avaus- ja sulkunapit pala ollenkaan ilman direct-I/O -toimintoa. Hissi-CPU ei lähetä käskyjä näille valoille, vaan jättää niiden ohjauksen koripaneelin Neuron-piirin tehtäväksi.

Käyttäessäsi valmista 2. vaiheen ohjelmaa voit todeta, että käskyjen ylläpito nopean painalluksen yhteydessä olisi tarpeellista: Nyt hissi-CPU ei aina havaitse koko painallusta, ja napin valo vain välähtää. Tämä puute korjataan työn seuraavassa vaiheessa.

3. VAIHE (vapaaehtoinen, 3 p): Lisätään edellisessä vaiheessa kehitettyyn ohjelmaan napeilla annettujen käskyjen pito-ominaisuus. Mikäli napin painallus kestää vähemmän kuin määritellyn minimiajan (200 ms), ylläpidetään käskyä kuvan 5. mukaisesti kunnes minimiaika täyttyy. Tällöin napin nopeakin hipaisu riittää ylläpitämään korikutsua kyllin pitkään, jotta hissi-CPU ehtii havaita sen. Verkkomuuttujaan kirjoitettu korikutsu ei näet siirry hissi-CPU:lle välittömästi, vaan hissi-CPU kysyy verkkomuuttujan arvoa säännöllisesti (polling). Käytännössä riittävä pitoaika on 200 ms, mutta havainnollisuuden vuoksi valmistaudu esittelemään ohjelmasi toimintaa myös pidemmällä pitoajalla, esim. 500 ms ja 2000 ms. Kuten kuvasta 6. havaitaan, ei käskyä ole tarkoitus ylläpitää tarpeettomasti, mikäli nappia painetaan muutenkin riittävän pitkään.



Kuva 5. Nappia painetaan alle 200 ms, jolloin kutsua ylläpidetään tarvittava aika.



Kuva 6. Nappia painetaan yli 200 ms, jolloin kutsua ei tarvitse erikseen ylläpitää.

YHDEN PISTEEN SUORITUS: Painettaessa mitä hyvänsä yhtä nappia pysyy napin valo ja verkkomuuttujaan kirjoitettu arvo yllä kuvien 5. ja 6. mukaisesti. Lisäksi niiden kerrosten valot palavat, joihin hissi-CPU on vastaanottanut korikutsun. Sovellus ei kuitenkaan kykene muodostamaan viiveitä useammalle nopealle peräkkäiselle painallukselle.

Valmistaudu lisäksi esittelemään ohjelmasi korikutsujen pito-ominaisuuksia kokeella, jossa asetetaan pitkä pitoaika (2000 ms) ja nappien valoja ohjataan ainoastaan nvoButtonNv-verkkomuuttujan arvojen perusteella. Tällöin korikutsujen kestoa voidaan havainnoida nappien indikaattorivaloista ilman, että hissi-CPU:n lähettämät indikaattorivalokäskyt häiritsevät havainnointia.

KAHDEN PISTEEN SUORITUS: Sovellus kykenee muodostamaan sammumisviiveen useammalle nopealle peräkkäiselle tai/ja samanaikaiselle painallukselle. Viiveen laskemiseen käytetään yhtä ajastinta, jonka arvo asetetaan aina uudelleen uuden painalluksen myötä. Näin peräkkäiset painallukset niputetaan samaan ryhmään, joka kytkeytyy pois ajastimen umpeuduttua.

Hissi-CPU tulkitsee pitkittyneen korikutsun pohjaan juuttuneeksi napiksi ja esittää häiriöilmoituksen näytöllään, mikäli kutsu on päällä yhtäjaksoisesti yli 60 sekuntia. Häiriöilmoitus kuitenkin poistuu välittömästi korikutsun antamisen päättyessä ja hissi-CPU toimii normaalisti.

KOLMEN PISTEEN SUORITUS: Sovellus muodostaa kaksi ryhmää poiskytkemisviiveiden laskemiseksi. Ensimmäinen yhden tai useamman napin samanaikainen painallus muodostaa ensimmäisen viiveryhmän, jonka valot sammuvat ensimmäisen ajastimen umpeutuessa. Kaikki myöhemmät painallukset sijoitetaan toiseen viiveryhmään, joka sammuu toisen ajastimen umpeutuessa. Ensimmäisen ajastimen/ryhmän vapauduttua kerätään vuorostaan siihen kaikki painallukset, kunnes toinen ajastin/ryhmä vapautuu.

Valmistaudu esittelemään ohjelmasi korikutsujen pito-ominaisuuksia kokeella, jossa asetetaan pitkä pitoaika (2000 ms) ja nappien valoja ohjataan ainoastaan nvoButtonNv-verkkomuuttujan arvojen perusteella. Painamalla kutsunappeja järjestyksessä (esim. 1-2-3-4...) voidaan tällöin tarkkailla, sammuvatko valot järjestyksessä tai edes jonkinlaisissa ryhmissä.

PITOLOGIIKKA-ESIMERKKI: Ylläpidettäviä nappikäskyjä säilytetään kahdessa vektorissa, ja nappiväylästä luettavat uudet arvot täydennetään näiden pitovektorien arvoilla ennen verkkomuuttujaan kirjoittamista. Binäärilogiikan (AND, OR, XOR) avulla selvitetään juuri ykköseksi muuttuneet bitit. Jos niitä on, tallennetaan ne vapaana olevaan pitovektoriin. Mikäli kummatkin pitovektorit ovat varattuna, täydennetään ylläpidettävät bittiarvot siihen pitovektoriin, joka varattiin viimeksi, ja pitovektorin varausaikaa jatketaan.

Varattua ylläpitovektoria käytetään aina määritellyn ylläpitoajan mukaisesti (esim. 200 ms) ja tätä aikaa kontrolloidaan joko vektorikohtaisella ajastimella tai vertaamalla nappiväylän 50 ms välein tapahtuvaan lukemiseen.

TÄMÄ ON VAIN ERÄS TAPA RATKAISTA TEHTÄVÄ, käytä omaa luovuuttasi.

BINÄÄRILOGIIKKA JUURI PAINETTUJEN NAPPIEN LÖYTÄMISEKSI

Ensin selvitetään uuden ja vanhan asentotiedon perusteella, onko napin asento juuri vaihtunut. Mikäli asento on juuri vaihtunut ja sen uusi tila on yksi, aktivoidaan ulostulo. Etsi oikeat logiikkaoperaattorit!

nappien uudet asennot	1001 0110 1001 0110 1011
nappien vanhat asennot	0110 1001 0101 0101 1011
asentoaan muuttaneet napit	1111 1111 1100 0011 0000
asentoaan muuttaneet napit	1111 1111 1100 0011 0000
nappien uudet asennot	1001 0110 1001 0110 1011
juuri ykköseksi muuttuneet napit	1001 0110 1000 0010 0000

HUOMIO! Tähän logiikkaan kannattaa tutustua heti jo tässä harjoitustyössä, sillä tätä käytetään myös seuraavan harjoitustyön vapaaehtoisessa osuudessa!

Ajastimien jäljellä oleviin aika-arvoihin kannattaa viitata varoen muualla kuin kyseisen ajastimen omassa ”when(timer_expires( ))”-osiossa. Ajastimen aika umpeutuu todennäköisesti jotakin toista ohjelman osiota suorittaessa, ja ajastimen omaa ”when(timer_expires( ))”-osiota päästään ajamaan vasta viiveellä ajastimen todellisen umpeutumisen jälkeen. Tarvittaessa voidaan käyttää boolean-muuttujaa, joka kertoo onko ylläpidettävät bittiarvot sisältävä vektori vielä varattuna. Tällöin pitovektori merkitään vapaana olevaksi vasta sen voimassaoloaikaa säätelevän ajastimen omassa ”when(timer_expires( ))”-osiossa.

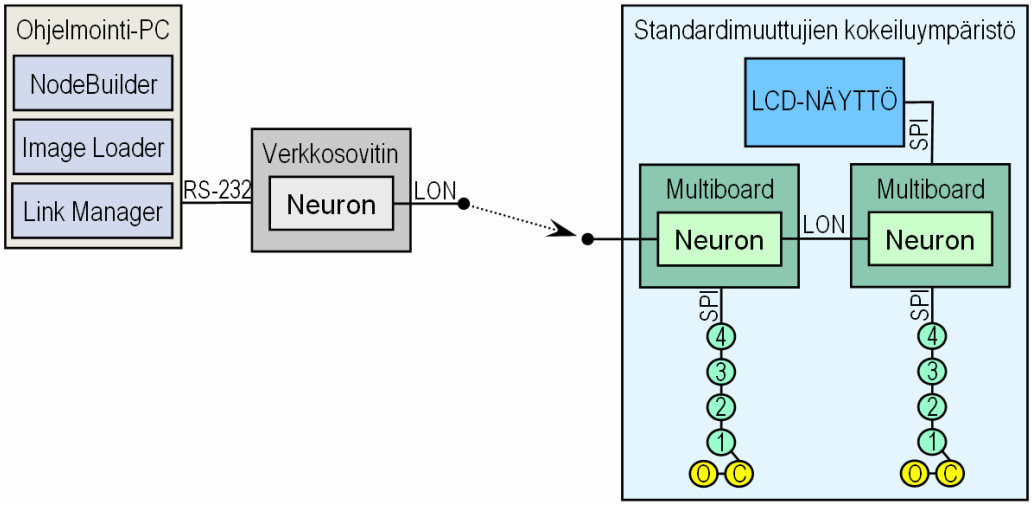
Liite 3. HARJOITUS: Sovellusviestit

Työkurssin kolmannessa harjoitustyössä tutustutaan sovellusviesteihin, joiden muodostamisesta ja lähettämisestä sovellusohjelman kirjoittaja huolehtii. Sovellusviesteihin siirrytään verkkomuuttujia käyttävää sovellusta muokkaamalla, jolloin havainnollisesti nähdään näiden kahden tiedonsiirtomenetelmän erot.

Verkkomuuttujia käyttävä sovellus puolestaan muokataan edellisen harjoitustyön sovelluksesta, joka nyt sovitetaan uuteen toimintaympäristöön sen verkkoliikenteen rajapintaa muokkaamalla. Samalla kerrataan verkkoliikenteen asetusten tekoa, johon edellisessä harjoituksessa jo tutustuttiin.

Edellisen harjoitustyön hissin korikutsuja antanut sovellus siirretään siis nyt kuvan 1. laitteistoon. Painonappien tietoja välittävät sisään- ja ulostuloverkkomuuttujat muokataan keskenään yhteensopiviksi. Näin sama sovellus voidaan sijoittaa laitteiston molempiin Neuron-piireihin, jotka lähettävät viestejä toisilleen. Kun toisessa solmussa painetaan nappia, syttyvät indikaattorivalot molempien solmujen napeissa. Solmun oman napin valo syttyy edellisessä harjoituksessa toteutetun direct-I/O -toiminnon ansiosta, ja toisen solmun valo Lon-verkon viestin ansiosta. Tämä viestiminen toteutetaan ensin verkkomuuttujalla ja sitten sovellusviestinä.

Lopuksi sovellusviestin lähetysmekanismia hiotaan paremmaksi, jolloin yhtä tapahtumaa kohti lähetetään oikeaoppisesti vain yksi viesti. Tämän havainnollistamiseksi solmuihin tehdään tarvittavat muutokset.



Kuva 1. Harjoitustyössä käytettävä laitteisto.

Tutustu Neuron C:n sovellusviesteihin seuraavia dokumentteja käyttäen:

- ”Neuron C -ohjelmointi”, luku 4. työkurssin valmistelleesta diplomityöstä. Erityisesti sovellusviestit.

- ”Neuron C Programmers Guide”, yleinen Neuron C -ohjelmointiopas. Lue kuudetta lukua sivulle 17. saakka.

- ”Neuron C Reference Guide”, hakuteoksena käytettäväksi. Dokumentti sisältää ohjeet Neuron-piirin varusohjelmiston funktioiden käyttöön.

1. VAIHE (pakollinen): Tässä harjoitustyössä muokataan edellisessä harjoitustyössä kehitettyä ohjelmakoodia. Muokkaus kohdistuu edellisessä harjoituksessa annetun koodipohjan rakenteisiin, joita muuttamalla sovellus voidaan sijoittaa uuteen ympäristöön. Näin edellisestä harjoituksesta tutun sovelluksen eteenpäin kehittämistä voidaan jatkaa ja samalla tutustua LonTalk-protokollan tarjoamiin erilaisiin viestintämenetelmiin.

Muokkaa edellisessä harjoituksessa kehittämääsi ohjelmakoodia seuraavasti:

1. Poista kaikki muut tyyppimäärittelyt lukuun ottamatta struct-määrittelyjä ”T_indicator_nv” ja ”T_button_nv”, joilla määritellään painonappien asento- ja valotietoja välittävät verkkomuuttujat. Poista vielä indikaattorivalotiedot vastaanottavan verkkomuuttujan struct-määrittelystä ”T_indicator_nv” kuittaustietokenttä ”ack”. Näin painonappien tietoja välittävät sisään- ja ulostuloverkkomuuttujat saadaan keskenään yhteensopiviksi.

2. Poista vakio ”S_node_setup”. Lyhennä verkkomuuttujien valintatunnukset sisältävän vakiovektorin ”C_nv_idxs” alkioiden määrä viidestä kahteen. Jätä jäljelle valitsimet verkkomuuttujille ”nviIndicatorNv” sekä ”nvoButtonNv”.

3. Poista I/O-objektien “ioDisplay” ja “ioDisplaySelect” määrittelyt sekä globaali verkkomuuttujavektori ”DisplayBuffer”. Poista muut verkkomuuttujat paitsi ”nviIndicatorNv” sekä ”nvoButtonNv”. Poista verkkomuuttujan ”nviNodeSetupNv” alustava memcpy-lause resetoinnista.

4. Poista verkkomuuttujan ”nviIndicatorNv” päivittymiseen reagoivasta when-lauseesta viittaukset verkkomuuttujan ack-kenttään tai ulostuloverkkomuuttujaan ”nvoOutputAckNv”, koska näitä ei tässä vaiheessa enää ole olemassa. Mikäli when-lause jää muuten tyhjäksi, jätä sinne yksi puolipiste ”;”. Poista myös verkkomuuttujiin ”nviLocalOutputNv” ja ”nviNodeSetupNv” reagoivat when-lauseet kokonaan, koska näitäkään verkkomuuttujia ei enää ole.

Tämän jälkeen käytettävissä on sovellusohjelma, jota voidaan käyttää sekä nappien asentotietojen lähettämiseen että vastaanottamiseen. Sama sovellus ladataan molempiin kuvan 1. laitteiston Neuron-piireihin, jolloin solmun painonappien merkkivaloista nähdään viereisessä solmussa suoritetut painallukset. Viestinnän onnistuminen vaatii enää sopivien osoitetietojen ja verkkomuuttujien valitsinten asettamisen.

Aseta vasemman puoleisen Neuronin omaksi verkko-osoitteeksi 0.3.1 (pääverkko.aliverkko.solmu) ja kohdeosoitteeksi solmu 0.3.2. Myöhemmin oikean puoleisen Neuronin sovellusohjelmaan asetetaan sen omaksi verkko-osoitteeksi 0.3.2 ja kohdeosoitteeksi 0.3.1.

Aseta myös 1. solmun ulostuloverkkomuuttujalle valitsimeksi 0x12 ja sisääntuloverkkomuuttujalle valitsimeksi 0x21. Vastaavasti myöhemmin oikean puoleisen Neuronin sovellusohjelmaan asetetaan ulostuloverkkomuuttujalle valitsimeksi 0x21 ja sisääntuloverkkomuuttujalle valitsimeksi 0x12. Kaikilla loogisesti yhteen kytketyillä verkkomuuttujilla tulee olla sama valitsin, jonka perusteella vastaanottaja osaa sijoittaa viestipaketin oikeaan sisääntuloverkkomuuttujaansa.

Lataa sovellus vasemman puoleiseen Neuroniin ja testaa sen toimintaa. Nappien valojen tulisi nyt reagoida painalluksiin edellisessä harjoitustyössä kehittämäsi toiminnallisuuden mukaisesti. Pohjassa olevien nappien valojen tulisi palaa direct-I/O

-toiminnon ansiosta, ja painallusten jälkeen valojen tulisi pysyä päällä kehitettyjen ylläpito-ominaisuuksien mukaisesti. Mikäli sovellusohjelma toimii odotetusti, tee siihen tarvittavat osoitemuutokset ja lataa se vielä oikean puoleiseen piiriin. Nyt voit kokeilla solmujen välistä yhteistoimintaa. Mikäli viesti ei siirry toiselle solmulle, tarkista verkkoasetukset. Voit tarkastaa Neuron-piirien sisältämät verkkoasetukset Lon Node Image Loader -ohjelmalla edellisen harjoituksen kuvan 3. mukaisesti. Solmun oma osoite löytyy Domains-välilehdestä, kohdeosoite Address-välilehdestä ja verkkomuuttujien valitsimet NVs-välilehdestä.

2. VAIHE (pakollinen): Seuraavaksi siirrytään solmujen välisessä Lon-verkon viestinnässä verkkomuuttujista sovellusviestien käyttöön. Tämä tehdään hyvin konkreettisesti muuttamalla verkkomuuttujat globaaleiksi muuttujiksi eli puskureiksi, joista tieto välitetään Lon-verkkoon sovellusviestien rakenteita käyttäen. Osoitteet säilyvät muuten ennallaan, mutta verkkomuuttujien asetuksia ei enää tarvita.

> Poista verkkomuuttujien valintatunnukset sisältävä vakiovektori ”C_nv_idxs”, nv_struct-tyypin globaali muuttuja ”m_stNv” sekä valintatunnukset kirjoittava for-silmukka resetoinnista. Muuta verkkomuuttujat ”nviIndicatorNv” sekä ”nvoButtonNv” globaaleiksi muuttujiksi poistamalla niiden määrittelyistä termit ”network output/input” sekä mahdollinen bind_info( )-määre. Poista myös when-lause, jota ”nviIndicatorNv” ohjasi.

Nappiväylän ajastimen ohjaamaan tehtävään lisätään sovellusviestin lähetys, joka kirjoittaa globaaliksi muuttujaksi muutetun entisen verkkomuuttujan arvoa verkkoon.

> Luo sovellusviesteissäsi käyttämäsi kohdeosoitteen tunnus eli ”tag” sekä viestisi sisällön vastaanottajalle määrittelevä viestin tunnuskoodi. Suorita sovellusviestin lähetys nappiväylää lukevassa when-lauseessa määrittelemiäsi tunnuksia käyttäen. Kopioi data globaalista muuttujasta viestiobjektiin tavu kerrallaan kopioimalla. Turvallisuussyistä älä käytä varusohjelmiston funktiota ”memcpy( )”. Tämän tehtävän toteuttaminen vaatii vain kahden tavun siirtämistä sovellusviestin datakentässä. Vaikka tiedonsiirrossa puskureina toimivien globaalien muuttujien pituus on 30 tavua, on niissä tämän harjoituksen kannalta merkitsevää dataa vain kahdessa tavussa. Etsi lähtevän tiedon puskurista nämä kaksi tavua ja kirjoita ne sovellusviestin datakenttään.

Sovellusviestin vastaanottaminen luodaan omaksi tapahtumakseen, joka päivittää globaaliksi muuttujaksi muutetun entisen verkkomuuttujan arvoa.

> Luo viestin saapumiseen reagoiva when-lause, jonka liipaisuehtona on määrittelemäsi viestin numeerinen tunnuskoodi. Kopioi viestiobjektin datakentän sisältämät kaksi tavua oikeisiin kohtiin puskurimuistina toimivaan globaaliin verkkomuuttujaan. Lisää kopiointia edeltävä saapuneen viestin koon tarkastus. Tällöin kopiointi suoritetaan vain, mikäli saapuneen viestin datakentän pituus on kaksi.

Lopuksi lisätään tehtävänajoitusta ohjaavan vuorottimen nollaus sekä yleinen viestien saapumistapahtuman tarkastava when-lause. Tehtävät järjestetään ohjelmakoodin sisällä siten, ettei ajastimen ohjaava when-lause vie kaikkea sovellusprosessorin huomiota. Nämä toteutetaan työkurssin valmistelleen diplomityön luvun 4. mukaisesti.

Sovellusviesteillä toteutetun version tulisi toteuttaa sama toiminnallisuus kuin edellisen verkkomuuttujilla toteutetun. Ainoastaan verkonhallintaohjelmalla voidaan todeta, että verkkomuuttujia koskevat listaukset ovat kadonneet NVs-välilehdestä. Solmun oman ja kohdeosoitteiden listausten voidaan huomata säilyneen entisellään.

3. VAIHE (vapaaehtoinen, 2 p): Sovellusviestin lähetysmekanismia hiotaan paremmaksi, jolloin yhtä tapahtumaa kohti lähetetään oikeaoppisesti vain yksi viesti. Tämän havainnollistamiseksi solmut jaetaan lähettävään ja vastaanottavaan yksikköön.

Harjoitustyön edellisessä vaiheessa nappien asentotietoja lähetettiin ajastimen ohjaamana säännöllisesti 50 ms välein. Kahden tällaisen solmun yhdistelmä on toimintakykyinen, sillä Neuron-solmut kykenevät käsittelemään 100 sanomaa sekunnissa. Tämä ei ole kuitenkaan oikeaoppinen tapa lähettää sovellusviestejä, sillä verkon kapasiteettia tuhlaantuu. Viestejä tulisi lähettää vain tarvittaessa, jolloin tiedonsiirtokanavan kapasiteetti riittäisi paljon suuremmalle määrälle solmuja.

Neuron C sisältää funktiot ”io_changes( )” ja ”io_update_occurs( )”, jotka on tarkoitettu liipaisemaan when-lause muutoksen tapahtuessa I/O-portissa. Tätä toimintamallia käyttäen harjoitustyön tehtävä olisi helppo toteuttaa oikeaoppisesti. Muista I/O-porteista poiketen SPI-väylä ei kuitenkaan ilmaise mahdollista sisääntuloarvon muutosta, vaan SPI-väylää tulee lukea säännöllisesti 50 ms välein.

Ongelman ratkaisemiseksi lähettävään solmuun kehitetään logiikka, joka eliminoi napin pohjassa olon vaikutuksen. Viesti lähetetään ainoastaan napin kytkeytymishetkellä, jolloin myös napin valo sytytetään. Tavoitteena on saada napin valo välähtämään vain kerran, vaikka nappia pidetään pohjassa. Tämä havainnollistaa, että viestejäkin lähtee vain yksi kappale.

Vastaanottavassa solmussa nappien valoihin rakennetaan ON- ja OFF-tilat, jotka myös havainnollistavat ainoastaan yhden viestin vastaanottamista. Mikäli viestejä vastaanotettaisiin enemmän, paljastuisi se napin valon välkkymisestä. Painettaessa siis lähettävässä solmussa nappia 1, vaihtuu vastaanottavassa solmussa napin 1 valon tila. Se syttyy jos se oli pois päältä, tai se sammuu jos se oli päällä. Kaikki muut napit toimivat samoin.

Hyödynnä lähettävän solmun logiikassa ”Korikutsut”-harjoituksessa neuvottua binäärilogiikkaa juuri painettujen nappien löytämiseksi. Voit käyttää edellisen vaiheen lähtevien viestien puskurimuistina toiminutta entistä ulostuloverkkomuuttujaa edelleen samaan tarkoitukseen. Nyt siihen kirjoitetaan napin fyysisen tilan sijasta napin looginen tila, joka on 1 vain juuri nappia painettaessa. Tämä looginen tila esitetään myös lähettävän solmun indikaattorivaloilla.

Vastaanottavan solmun indikaattorivalojen tila-arvoja voidaan ylläpitää edellisen vaiheen saapuvien viestien puskurimuistissa. Sovellusviestin saapuessa uudet tilat päivitetään puskurimuistiin sopivaa loogista operaattoria käyttämällä.

SUORITUS: Ryhmä varaa työskentelyajat simulointilaitteiston luona olevasta varauslistasta. Työaikoja saa varata korkeintaan kolmeksi tunniksi kerrallaan. Työtä suositellaan tehtäväksi määräaikoihin nähden etupainotteisesti, jotta vältytään ruuhkilta laitteiston käytössä.

Tehty harjoitus hyväksytetään assistentilla erikseen määrättyinä esittelyaikoina. Viimeisen onnistuneesti toteutetun vaiheen esittely riittää. Esittelyajat ja harjoituskohtaiset aikataululliset takarajat käyvät ilmi varauslistasta. Onnistuneen esittelyn jälkeen harjoitustyön viimeisen hyväksytyn vaiheen lähdekoodi lähetetään vielä assistentille sähköpostin liitetiedostona. Lähettämistä varten tiedosto nimetään muotoon ”A_B_C.nc”, jossa A = ryhmän numero, B = harjoitustyön numero ja C = vaiheen numero. Sähköposti nimetään muotoon ”Ryhmä A, harjoitustyö B” ja varustetaan jäsenten nimillä ja opiskelijanumeroilla.

ARVOSTELU: Kurssin harjoitustyöt koostuvat pakollisista sekä vapaaehtoisista osista. Seuraavaan harjoitustyöhön ei saa siirtyä, ellei edellisen harjoitustyön pakollista osaa ole suoritettu.

Kurssiarvosana määräytyy vapaaehtoisista tehtävistä kerättyjen pisteiden perusteella taulukon 1. mukaisesti. Suorittamalla kaikista kurssin harjoitustöistä pelkän pakollisen osuuden saa arvosanaksi ykkösen. Viitosen saadakseen tulee kerätä vähintään 80 % suorituspisteistä. Pisteet ilmoitetaan prosentteina, sillä jos jonkin kurssin harjoitustyön suoritus estyy teknisistä syistä, lasketaan suoritettujen pisteiden osuus käytännössä tarjolla olleista maksimipisteistä.

Taulukko 1. Kurssiarvosanan määräytyminen.

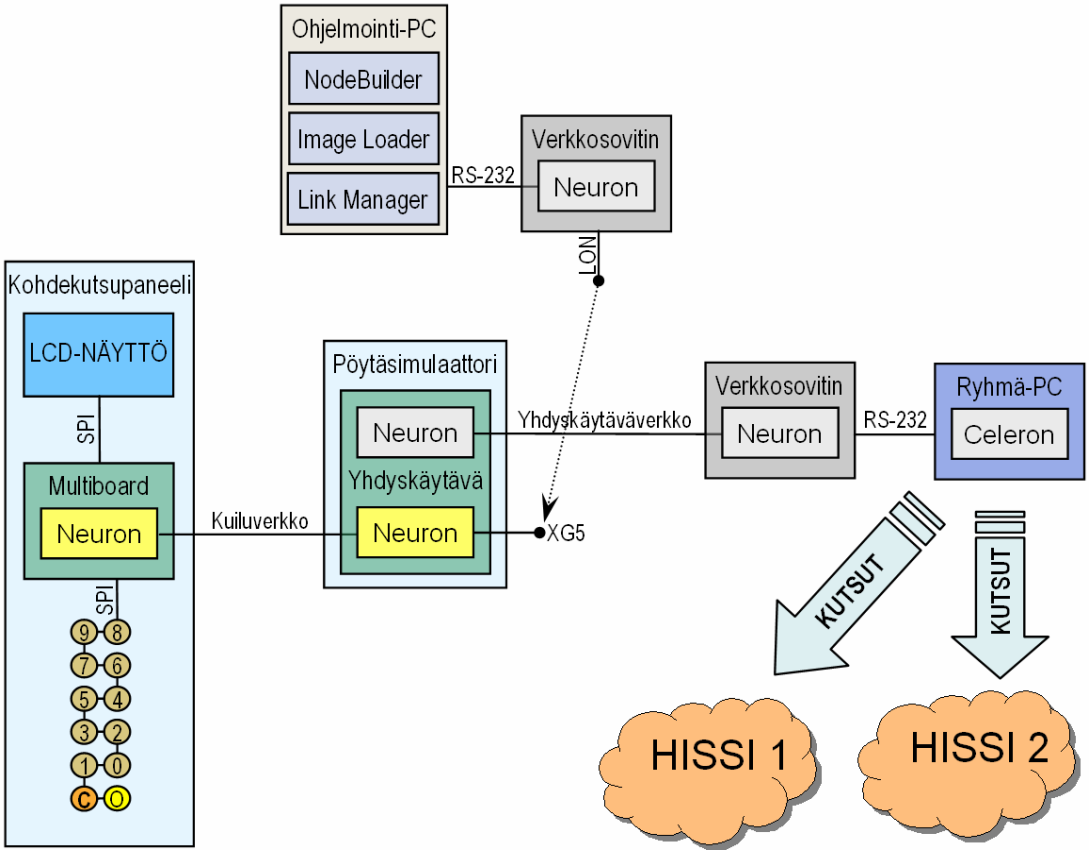
Arvosana	1	2	3	4	5
Pisteet	<20 %	≥20 %	≥40 %	≥60 %	≥80 %

Kolmannessa harjoitustyössä on kolme vaihetta. Niistä kaksi ensimmäistä ovat pakollisia, ja viimeisestä saa suorituspisteitä. Viimeisestä vaiheesta on mahdollista saada kaksi pistettä, mikäli tehtävänanto toteutuu kaikilta osin.

Liite 4. HARJOITUS: Kohdekutsut

Työkurssin neljännen harjoitustyön tavoitteena on tuottaa Neuron C -ohjelmakoodi hissin kohdekutsujen antamiseen. Kehitettävä sovellusohjelma ladataan kohdekutsupaneelin Neuron-piiriin, jossa se kääntää viestit Lon-kuiluverkon ja Neuron-piirin SPI-väylän välillä. Kohdekutsupaneelin napeilla annettu palvelupyyntö lähetetään sovellusviestinä ryhmä-PC:lle, jonka palauttama vastaus esitetään kohdekutsupaneelin LCD-näytöllä.

Kurssin käytössä oleva hissin toimintoja simuloiva laitteisto on esitelty kokonaisuudessaan ”Simulointiympäristön käyttöohjeessa”. Tämän työohjeen kuvassa 1. esitetään ne laitteiston osat, joiden kanssa harjoitustyössä ollaan välittömästi kosketuksissa.



Kuva 1. Harjoitustyössä käytettävä laitteisto.

Kohdekutsupaneeli sijoitetaan tyypillisesti rakennuksen aulaan, jonka kautta suurin osa hissin käyttäjistä kulkee. Aulasta käyttäjä tilaa hissin näppäilemällä kohdekutsupaneeliin sen kerroksen numeron, johon hän on matkalla. Tällöin ryhmä-PC valitsee hissiryhmästä suunnan ja etäisyyden perusteella sopivimman hissin palvelemaan tätä kutsua. Ryhmä-PC palauttaa kohdekutsupaneelille matkustajien kuljetustiedot, joilla matkustaja opastetaan oikean hissin ovelle.

Kohdekutsupaneelista lähetettävien viestien kohteena on ryhmä-PC. Viestit kulkevat kuitenkin yhdyskäytävän kautta, mikä huomioidaan viestien osoite- ja tunnustietojen määrittelyssä. Yhdyskäytävä jakaa Lon-verkon kanaviin ja reitittää kanavien väliset viestit.

Harjoitustyössä tuotettavaa ohjelmakoodia varten annetaan pohjatiedosto, jota käytetään sovelluskehityksen pohjana. Pohja sisältää lähes valmiin version yksinkertaisesta kohdekutsunanto-ohjelmasta. Pohja täydennetään toimivaksi lisäämällä siihen oikeat Lon-verkon osoitteet ja kohdekutsujen tunnuskoodit. Lisäksi luodaan ryhmä-PC:n lähettämiä viestejä vastaanottava when-lause, josta ohjataan kohdekutsupaneelin LCD-näyttöä.

Harjoitustyön vapaaehtoisessa osuudessa sovelluksen yksinkertaista toimintalogiikkaa kehitetään eteenpäin. Tällöin kutsunanto tapahtuu näppäinyhdistelmänä, ja näytön ominaisuuksia hyödynnetään täysipainoisesti kutsunannon vaiheiden viestittämiseksi käyttäjälle.

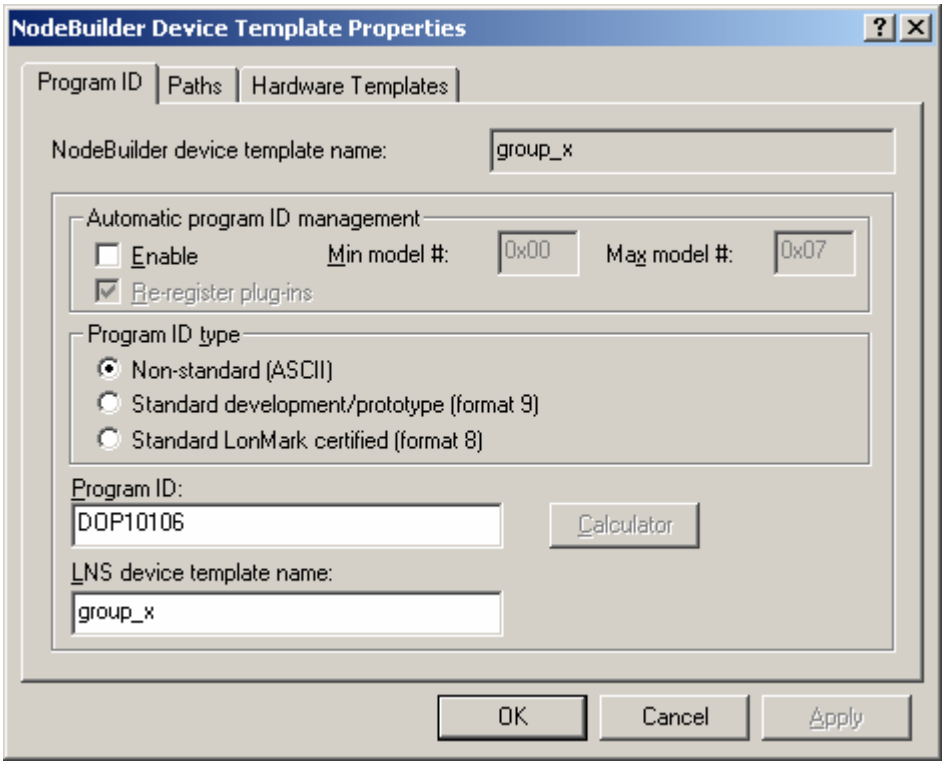
Tässä harjoituksessa simulointiympäristön Lon-verkkoon kytkeydytään vasemmanpuoleisen pöytäsimulaattorin liittimestä ”XG5”. Kohdekutsupaneeli on kytketty vasemman hissin kuiluverkkoon, joten sen Neuron-piiri tavoitetaan ainoastaan kytkeydyttäessä Lon-verkkoon vasemmanpuoleisen pöytäsimulaattorin liittimestä. Kertaa ”Simulointiympäristön käyttöohjeesta” simulointilaitteiston komponenttien toimintaa sekä Lon Node Image Loader -verkonhallintaohjelman käyttöä hissiverkon laajimmassa osassa.

Harjoitustyön teko voidaan aloittaa kytkemällä käyttöjännite ainoastaan vasemman puoleiseen pöytäsimulaattoriin, jolta myös kohdekutsupaneeli ja yhdyskäytävä saavat käyttöjännitteensä. Ryhmä-PC ja oikeanpuoleinen hissi voidaan käynnistää myöhemmin valmiin sovelluksen testaamiseksi. Ryhmä-PC käynnistetään aina viimeisenä ja sammutetaan ensimmäisenä. Sen käynnistyksen yhteydessä on odotettava, että virtakytkimen vieressä olevat LEDit ”LIFT 1” ja ”LIFT 2” syttyvät. Ne ovat 16 LEDin ryhmän kaksi ylintä LEDiä.

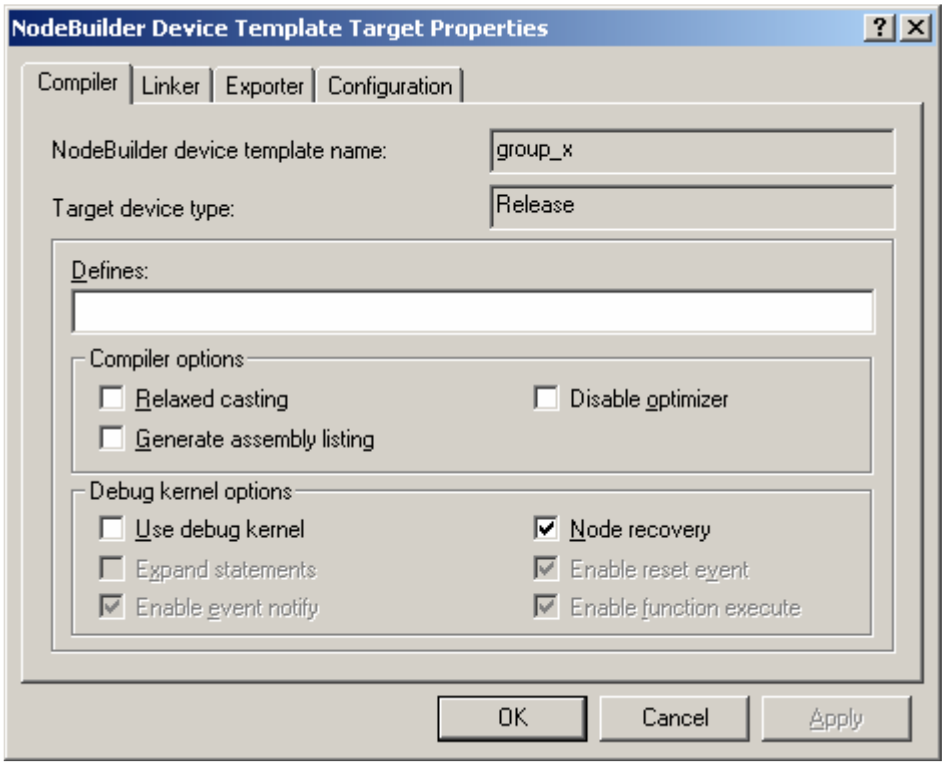
1. VAIHE (pakollinen): Käytä edellisissä harjoituksissa luomaasi NodeBuilder-projektia. Kopioi projektisi lähdekoodihakemistoon tiedosto “pohja_4.nc” ohjelmointi-PC:n hakemistosta ”D:\Mallit” ja nimeä se muotoon ”group_x.nc” (x = ryhmänne numero) alkuperäisen projektitiedoston korvaamiseksi.

Muuta laitemallin asetukset vastaamaan kuvaa 2. Asetukset löytyvät napsauttamalla NodeBuilderin Workspace-ikkunassa laitemallin tunnusta hiiren oikealla napilla ja valitsemalla avautuvasta valikosta ”Settings”. Tarkista, että laitemallin julkaisuversiossa on toiminto ”Node recovery” päällä kuvan 3. mukaisesti. Tarkista vielä, että laitteiston mallitiedoston (Hardware Template) asetukset vastaavat kuvaa 4.

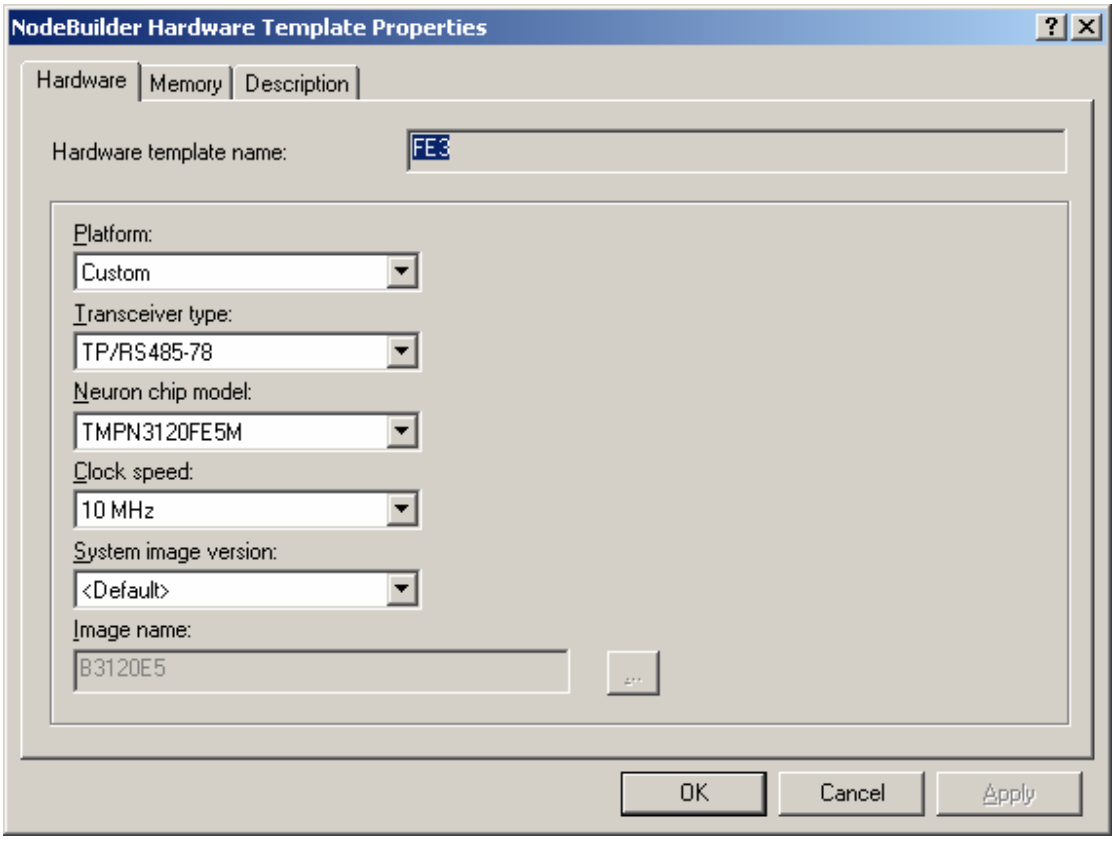
Kuvan 2. asetuksiin vaihdettiin sovellusohjelman tunnus, joka on osa NodeBuilderilla luotavaa Neuron-piirin sovellusohjelmaa. Lon Node Image Loader -ohjelmaa käytettäessä verkon solmut erotetaan toisistaan sovellusohjelman tunnuksien perusteella, eikä tunnuksia tule muuttaa uusien sovellusohjelmien versioiden myötä. Kohdekutsupaneelissa käytetään siis aina sovellusohjelman tunnusta DOP10106.



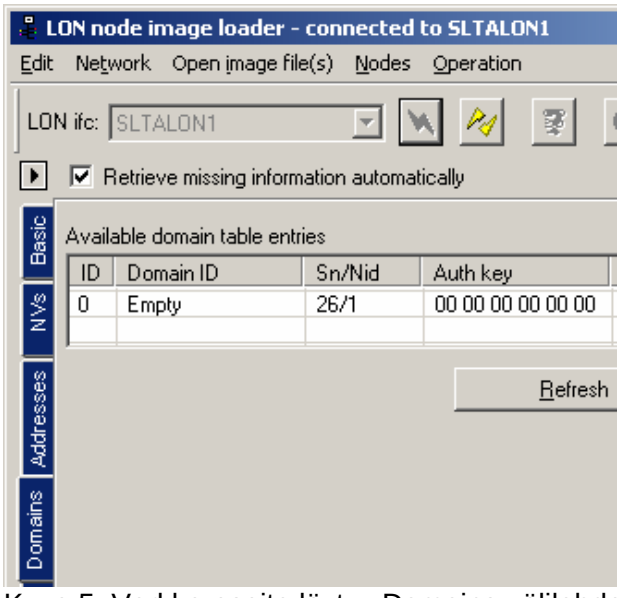
Kuva 2. Aseta laitemalliin sovellusohjelman tunnukseksi **DOP10106**.



Kuva 3. Tarkista, että laitemallin julkaisuversiossa on ”Node recovery” päällä.



Kuva 4. Laitteiston mallitiedoston (Hardware Template) asetukset.



Kuva 5. Verkko-osoite löytyy Domains-välilehdestä. Sn = subnet, Nid = node ID.

Annetussa pohjatiedostossa on kohdekutsupaneelin Neuronin omaksi verkko-osoitteeksi asetettu virheellisesti 0.2.3. (pääverkko.aliverkko.solmu). Kuva 5. esittää näkymän verkonhallintaohjelmaan tilanteessa, jossa kohdekutsupaneelin verkko-osoite on oikea. Pääverkon tunnuksessa merkintä ”empty” tarkoittaa, että tunnuksen pituus on määritelty nollaksi. Tällöin pääverkon tunnus myös tulkitaan nollaksi.

Tee tarvittavat korjaukset ohjelmakoodiisi, jotta solmusi osaa vastaanottaa yhdyskäytävän kautta saapuvat ryhmä-PC:n viestit. Tee korjaukset kuvan 5. perusteella, älä vielä lataa sovellustasi Neuron-piiriin.

Kohdekutsupaneelin lähettämien viestien kohdeosoitteeksi on myös virheellisesti määritelty 0.2.3 (pääverkko.aliverkko.solmu). Näiden viestien lopullinen kohde on ryhmä-PC, mutta lähtevän viestin kohdeosoitteeksi merkitään yhdyskäytävän verkko-osoite. Yhdyskäytävä reitittää viestin eteenpäin ryhmä-PC:lle.

Etsi yhdyskäytävä verkonhallintaohjelmaan automaattista solmujen etsintää käyttäen. Yhdyskäytävän tunnus on ”GTL10114”. Selvitä verkonhallinta-ohjelmalla yhdyskäytävälle määritelty verkko-osoite ja määrittele tämä kohdekutsupaneelista lähetettyjen viestien kohdeosoitteeksi ohjelmakoodiisi. Valitse kohdeosoitteeksi ensimmäinen yhdyskäytävän sisältämästä kahdesta verkkoaluetaulukon merkinnästä.

Jokainen sovellusviestejä käsittelevä Neuron-sovellus suodattaa ja lajittelee vastaanottamiaan viestejä viestiobjektin code-kentän sisältämän tunnuskoodin avulla. Tätä tunnuskoodia voidaan käyttää valmiin tapahtumamäärittelyn ”msg_arrives( )” tarkentavana määreenä, jolloin ohjelmakoodin ajoa voidaan ohjata saapuneen viestin tunnuskoodin perusteella.

Code-kenttään voidaan määritellä vain rajallinen määrä numeerisia tunnuskoodeja. Koska käytettävissä olevien tunnuskoodien määrä ei ole riittänyt laajaan järjestelmään, on hissiverkon sovellusviesteissä otettu käyttöön toinen tunnuskoodeja sisältävä bittikenttä, joka sisältyy viestiobjektin data-kenttään.

Nämä data-kentän tunnukset toimivat tarkentavina alamääreinä code-kentässä käytettäville yleisemmän tason tunnuksille. Vastaanotetut viestit lajitellaan siis ensin code-tunnuksen perusteella, jonka jälkeen tarkastetaan viestin data-kenttään sisältyvä tunnus. Harjoitustyössä käytettävät tunnukset on määritelty annettavaan koodipohjaan.

Kohdekutsupaneelista lähetettyjen viestien lopullinen määränpää on ryhmä-PC, mutta viestit kulkevat yhdyskäytävän kautta. Lähetettävän viestipaketin Lon-verkon osoite haetaan Neuron-piirin osoitetaulukosta kohdeosoitteen tunnuksen (tag) perusteella, ja osoitetaulukkoon asetettiin edellä yhdyskäytävän verkko-osoite.

Kohdekutsupaneelista lähetettävän viestin code-kenttään määritellään tunnus, jonka perusteella yhdyskäytävä osaa käsitellä viestiä. Tämä eroaa tunnuksesta, joka code-kentässä tulee olla viestin lopulta saapuessa ryhmä-PC:lle. Ongelma on ratkaistu siten, että kohdekutsupaneelista lähetettävän viestin data-kenttä sisältää tarkentavan alamääretunnuksen lisäksi ryhmä-PC:lle tarkoitetun yleisen tason tunnuksen. Reitittäessään viestiä yhdyskäytävä kirjoittaa tuon yleisen tason tunnuksen ryhmä-PC:lle lähettämänsä viestin code-kenttään.

Harjoitustyössä kehitettävän sovellusohjelman rungoksi annettu pohjatiedosto sisältää valmiit rakenteet sovellusviestien lähettämiseksi. Täydennä niihin oikeat tunnuskoodit koodipohjan sisältämien vihjeiden ja määrittelyjen perusteella.

Kun ryhmä-PC vastaanottaa kohdekutsun onnistuneesti, palauttaa se kohdekutsupaneelille tiedot kutsua palvelevasta hissistä. Ryhmä-PC:n palauttama viesti sisältää kutsua palvelevan hissin tunnuksen (A tai B) sekä kohdekerroksen, johon tämä hissi kulkee. Hissi tulee siis ensin hakemaan matkustajia ensimmäisestä kerroksesta, jossa kutsupaneeli sijaitsee, ja jatkaa sitten kohdekerrokseen.

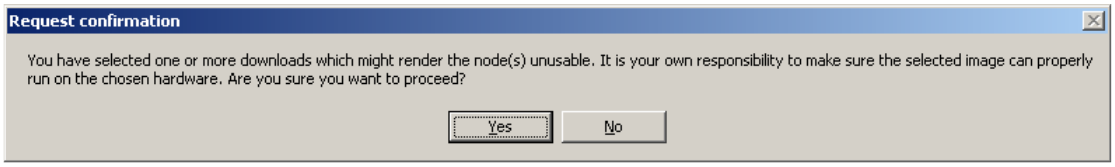
Muodosta sovellusohjelmaan when-lause, joka käsittelee ryhmä-PC:ltä vastaanotetun viestin. Esitä palvelevan hissin tiedot kohdekutsupaneelin LCD-näytöllä pohjatiedostossa annettua funktiota käyttäen.

Saapuneen viestin tunnuskoodeja tarkastaessasi huomioi, etteivät kohdekutsupaneelille saapuneet viestit enää sisällä yhdyskäytävälle tarkoitettua dataa, vaan yhdyskäytävä on suodattanut ne pois.

Kun sovellusohjelmasi on valmis kohdepiirille ladattavaksi, huomioi seuraavat asiat:

- Yhdyskäytävä on verkonhallintaohjelman ikkunassa aktiivisessa tilassa, vaikka siihen ei ole tarkoitus koskaan ladata mitään. ÄLÄ LATAA SOVELLUSOHJELMAA YHDYSKÄYTÄVÄN NEURON-PIIRIIN.

- Lon Node Image Loader -verkonhallintaohjelmaan ei ole ehditty sisällyttää tietoja kohdekutsunanto-ohjelman tunnuksesta ”DOP10106”. Tämän takia se antaa kuvan 6. esittämän varoituksen sovellusta piirille ladattaessa. Kysymyksen saa ohittaa vain, jos kuvien 2. - 4. asetukset ovat oikein asetettuina ja latauksen kohteena oleva piiri sisältää myös tunnuksen ”DOP10106”.



Kuva 6. Varmistuskysymys kohdekutsujen sovellusohjelmaa ladattaessa.

Toimivassa sovelluksessa kutsunapit 2 - 9 toimivat odotetusti. Napilla 0 hissi kutsutaan kuitenkin kerrokseen 10. ja napilla ”Cancel” kerrokseen 13. Nämä vastaavat nappien SPI-tunnuksia.

Napilla 1 hissiä kutsutaan samaan kerrokseen, jossa kutsupaneeli on. Tällöin ryhmä-PC ei välitä palvelupyyntöä eteenpäin. Ryhmä-PC ei myöskään palvele napilla ”Open” annettua kutsua. Tällöin hissi kutsutaan kerrokseen 113., mikä ylittää järjestelmään määritellyn maksimikerrosmäärän 40. Vaikka ryhmä-PC ei välitä palvelupyyntöä eteenpäin, se kuitenkin palauttaa viestin kohdekutsupaneelille. Tämä viesti sisältää pyydetyn kohdekerroksen numeron, mutta ei valitun hissin tunnusta. Tämän viestin perusteella voidaan kontrolloida, että viestit ovat kulkeneet molempiin suuntaan.

2. VAIHE (vapaaehtoinen, 3 p): Simulointilaitteistoon on määritelty kerrosten maksimimääräksi 40. Jotta kohdekutsu voidaan antaa kaikkiin mahdollisiin kerroksiin, on sovellusohjelmaan rakennettava toimintalogiikka näppäinyhdistelmänä syötetyn kutsun vastaanottamiseksi. Kohdekutsupaneelin LCD-näyttö mahdollistaa myös kutsunannon eri vaiheiden viestittämisen käyttäjälle.

YHDEN PISTEEN SUORITUS: Mikäli kahden pisteen suorituksen ehtoja (ks. alla) ei täytetä, mutta kutsuohjelmalla voidaan helposti tilata hissi kaikkiin kerroksiin 2. - 40., annetaan suorituksesta yksi piste. Kutsu on kyettävä antamaan korkeintaan kolmella napin painalluksella kaikkiin kerroksiin. LCD-näytön tulee kyetä näyttämään ryhmä-PC:n osoittama palveleva hissi (A tai B) sekä kohdekerros.

VINKKEJÄ: Suorita kaikki kohdekutsuviestien lähetykset keskitetysti omassa when-lauseessaan, johon vain asetat monipuoliset liipaisuehdot. Liipaisu voidaan tehdä rinnakkaisesti ajastimilla ja boolean-muuttujilla, jolloin ehdot yhdistetään loogisella OR-operaattorilla ”||”:

```
when ( ( EHTO 1. )) || ( EHTO 2. ) ) {
```

When-lauseita voidaan myös yhdistää rinnakkaisten liipaisuehtojen luomiseksi:

```
when ( EHTO 1. )
when ( EHTO 2. ) {
        …ohjelmakoodia…
} // end of WHEN
```

Suorita nappien painallusten käsittely (muuttujan ”button_pressed” muutokset) keskitetysti omassa when-lauseessaan, johon muodostat logiikan näppäinyhdistelmien vastaanottamiseksi. Tässä when-lauseessa voidaan asettaa ajastimia ja boolean-muuttujia muiden when-lauseiden ohjaamiseksi.

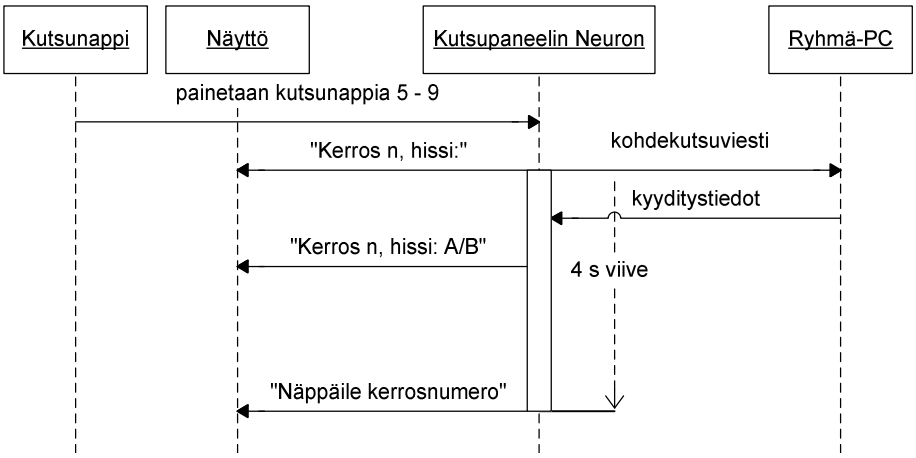
Määrittele lisäksi omia ajastimiasi ja niiden umpeutumiseen reagoivia when-lauseita tarpeen mukaisesti.

KAHDEN PISTEEN SUORITUS: Kohdekerroksen numero syötetään kuvien 7. - 10. mukaisesti yhdellä tai tarvittaessa kahdella painalluksella. Kutsun lähetys suoritetaan välittömästi ensimmäisen painalluksen jälkeen, mikäli toista numeroa ei ensimmäisen painalluksen jälkeen enää sallita. Tämä on esitetty kuvissa 7. ja 8. Simulointiympäristöön määritellyt sallitut kohdekerrokset ovat 2. - 40.

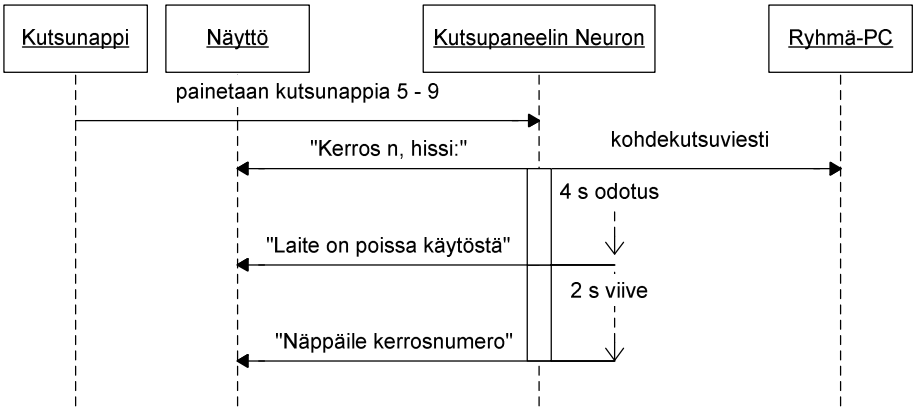
Mikäli ensimmäisen painalluksen jälkeen voidaan hyväksyä vielä toinen numero, odotetaan sen syöttämistä kolme sekuntia. Jos tämä aika ehtii umpeutua, lähetetään viesti ensimmäisen painalluksen mukaiseen kohdekerrokseen kuvan 9. mukaisesti. Jos ennen kolmen sekunnin umpeutumista vastaanotetaan toinen kutsunapin painallus, lisätään kohdekerroksen numeroon toinen numero ja lähetetään kohdekutsu välittömästi kuvan 10. mukaisesti

Kohdekerroksen numero esitetään näytöllä kohdekutsuviestin lähetyksen yhteydessä. Kohdekutsuviesti on voimassa neljä sekuntia, ja tämän ajan sisällä näyttöön päivitetään ryhmä-PC:n kyyditystietojen mukainen kohdekerros ja palvelevan hissin tunnus. Mikäli kyyditystiedot sisältävää sovellusviestiä ei vastaanoteta tämän ajan sisällä, esitetään näytöllä virheilmoitus kahden sekunnin ajan kuvan 8. mukaisesti.

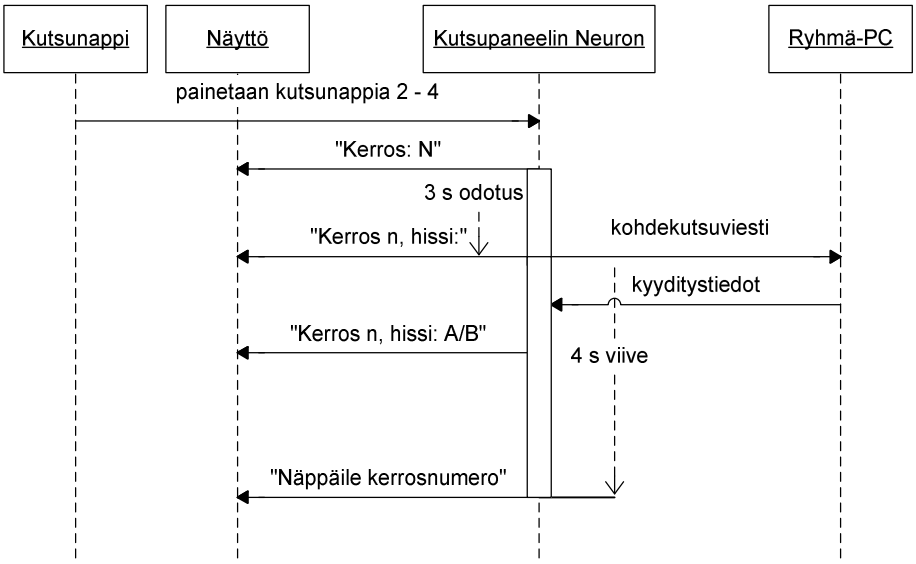
Näyttöön valmiiksi ohjelmoitujen virheilmoitusten käyttöä neuvotaan pohjatiedoston funktion ”display_on( )” määrittelyssä. Kuvien 7. - 16. näytön ilmoitusteksteissä kerrosnumeron fontti määrää numeron sijainnin näytöllä. Pikkukirjaimilla (esim. n ja nm) merkitty kerrosnumero esitetään näytön yläosassa pienillä numeroilla. Isoin kirjaimin merkitty kerrosnumero tai palvelevan hissin tunnus (A tai B) esitetään suurena symbolina LCD-näytön alaosassa.



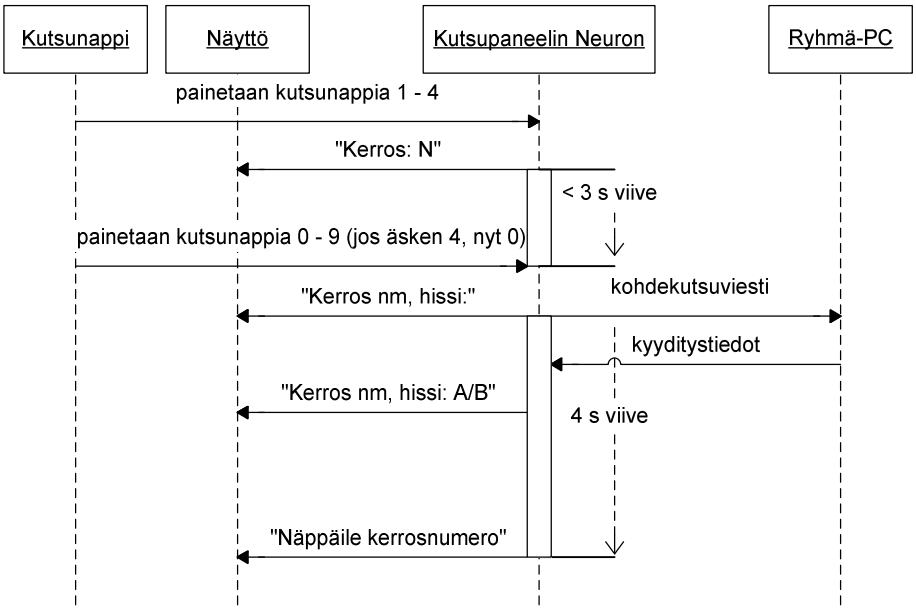
Kuva 7. Onnistunut kohdekutsunanto kerroksiin 5. - 9.



Kuva 8. Kohdekutsunanto kerroksiin 5. - 9., kun ryhmä-PC ei vastaa.



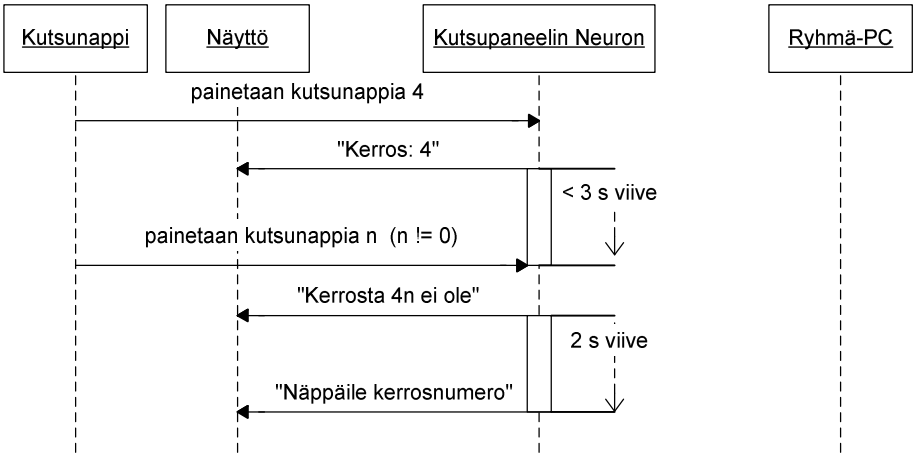
Kuva 9. Onnistunut kohdekutsunanto kerroksiin 2. - 4.



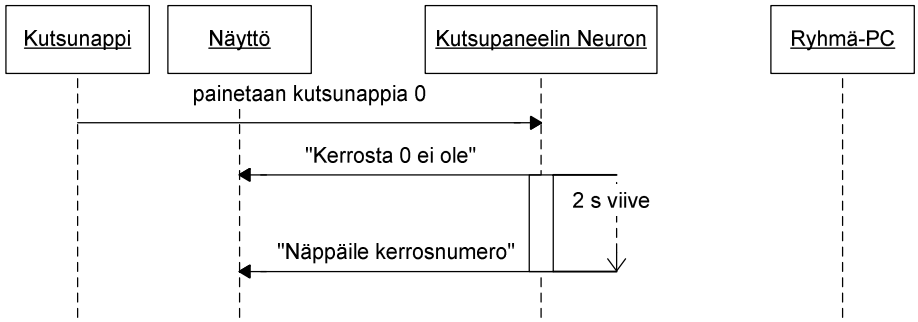
Kuva 10. Onnistunut kohdekutsunanto kerroksiin 10. - 40.

KOLMEN PISTEEN SUORITUS: Kutsunanto suoritetaan kuten kahden pisteen suorituksessa, mutta nyt huomioidaan myös erilaiset kutsunannon häiriötilanteet. Kaikki häiriötilanteet käsitellään harjoitustyössä paikallisesti, eikä häiriöilmoituksia vastaanoteta ryhmä-PC:ltä.

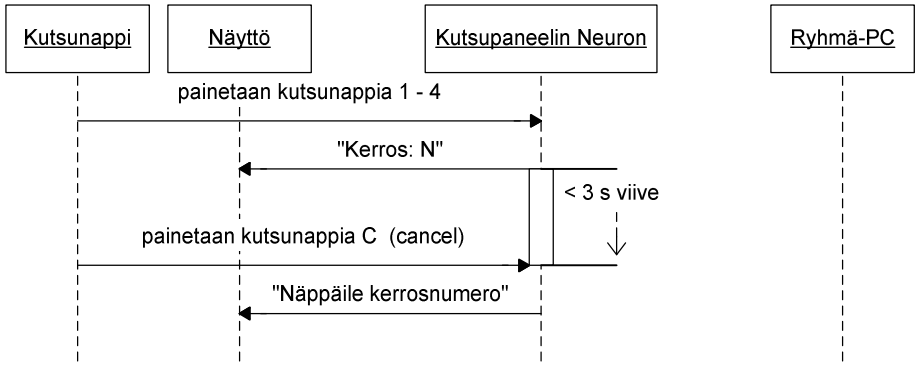
Toteuta kuvien 11. - 16. esittämät toiminnot kutsuohjelmassasi. **Virhetilan kahden sekunnin viiveen aikana ohjelman ei tule reagoida näppäinpainalluksiin.**



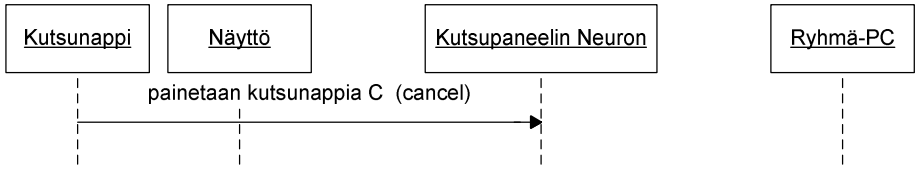
Kuva 11. Kohdekutsunanto kerroksiin 41. - 49., kun kerroksia on 40.



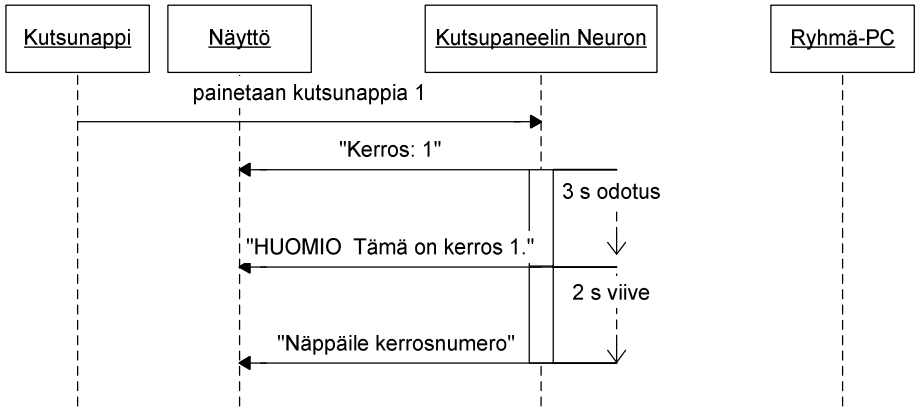
Kuva 12. Hissin tilaus kerrokseen 0.



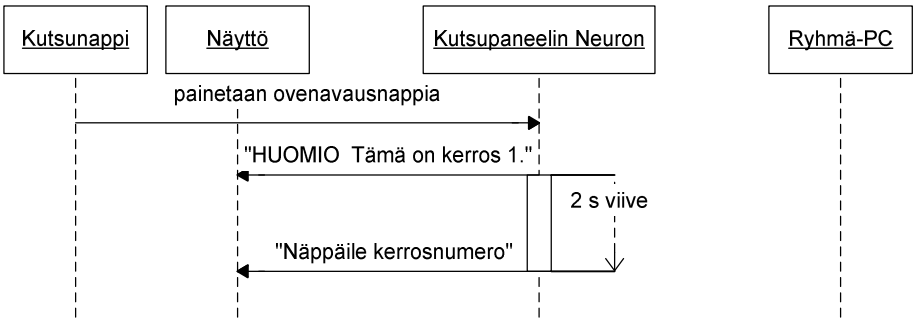
Kuva 13. Kutsun peruminen, kun on ensin painettu kutsunappia 1. - 4.



Kuva 14. Perustilassa C-napin painaminen ei aiheuta mitään reaktiota.



Kuva 15. Hissin tilaus 1. kerrokseen, kun paneeli sijaitsee 1. kerroksessa.



Kuva 16. Ovenavausnapin painaminen.

SUORITUS: Ryhmä varaa työskentelyajat simulointilaitteiston luona olevasta varauslistasta. Työaikoja saa varata korkeintaan kolmeksi tunniksi kerrallaan. Työtä suositellaan tehtäväksi määräaikoihin nähden etupainotteisesti, jotta vältytään ruuhkilta laitteiston käytössä.

Tehty harjoitus hyväksytetään assistentilla erikseen määrättyinä esittelyaikoina. Viimeisen onnistuneesti toteutetun vaiheen esittely riittää. Esittelyajat ja harjoituskohtaiset aikataululliset takarajat käyvät ilmi varauslistasta. Tämän jälkeen hyväksytysti esitellyn vaiheen lähdekoodi lähetetään vielä assistentille sähköpostin liitetiedostona. Lähettämistä varten tiedosto nimetään muotoon ”A_B_C.nc”, jossa A = ryhmän numero, B = harjoitustyön numero ja C = vaiheen numero. Sähköposti nimetään muotoon ”Ryhmä A, harjoitustyö B” ja varustetaan jäsenten nimillä ja opiskelijanumeroilla.

ARVOSTELU: Kurssin harjoitustyöt koostuvat pakollisista sekä vapaaehtoisista osista. Seuraavaan harjoitustyöhön ei saa siirtyä, ellei edellisen harjoitustyön pakollista osaa ole suoritettu.

Kurssiarvosana määräytyy vapaaehtoisista tehtävistä kerättyjen pisteiden perusteella taulukon 1. mukaisesti. Suorittamalla kaikista kurssin harjoitustöistä pelkän pakollisen osuuden saa arvosanaksi ykkösen. Viitosen saadakseen tulee kerätä vähintään 80 % suorituspisteistä. Pisteet ilmoitetaan prosentteina, sillä jos jonkin kurssin harjoitustyön suoritus estyy teknisistä syistä, lasketaan suoritettujen pisteiden osuus käytännössä tarjolla olleista maksimipisteistä. Vapaaehtoisten tehtävien lukumäärä vaihtelee eri harjoituksissa.

Taulukko 1. Kurssiarvosanan määräytyminen.

Arvosana	1	2	3	4	5
Pisteet	<20 %	≥20 %	≥40 %	≥60 %	≥80 %

Neljännessä harjoitustyössä on kaksi vaihetta. Niistä ensimmäinen on pakollinen ja jälkimmäisestä saa suorituspisteitä. Siitä on mahdollista saada jopa kolme pistettä, mikäli sovellusohjelma toteutetaan laajimman määrittelyn mukaisesti.

Liite 5. HARJOITUS: Standardit verkkomuuttujat

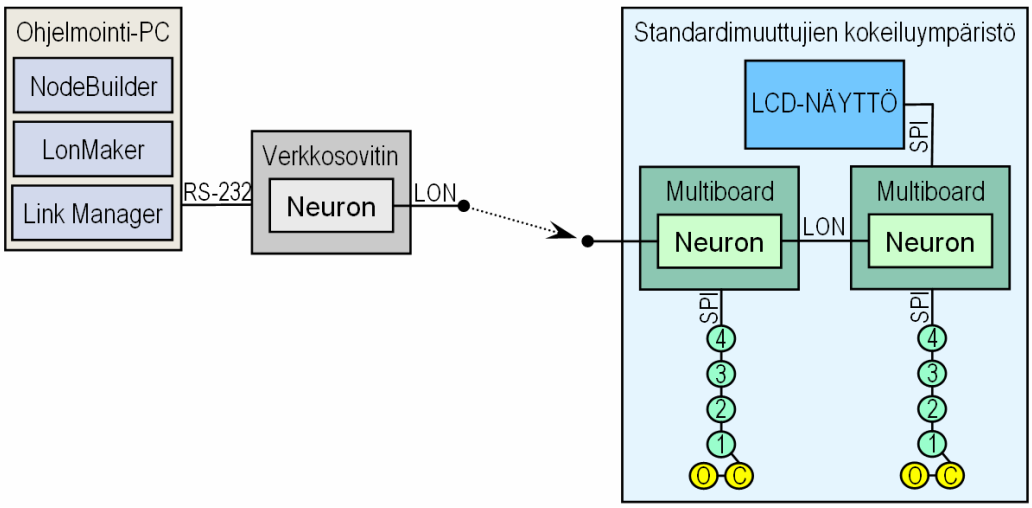
Työkurssin viimeisessä harjoitustyössä luodaan verkkoon kirjoittava sekä verkosta lukeva solmu, joiden välinen looginen kytkeminen suoritetaan LonMaker-verkonhallintaohjelman graafisella käyttöliittymällä. Solmut toteuttavat standardin verkkoliikenteen rajapinnan, joka generoidaan NodeBuilderin sisältämällä Code Wizard -ohjelmalla automaattisesti. Sovellusohjelman kirjoittajan tehtäväksi jää tällöin generoidun ohjelmarungon täydentäminen toimivaksi ohjelmaksi.

Molempiin harjoitustyön solmuihin luodaan kaksi toiminnallista lohkoa, jotka toteuttavat vaihtoehtoisen verkkoliikenteen rajapinnan. Haluttu toiminnallinen lohko kytketään käyttöön, kun sovellusohjelman sisältävää valmista laitetta asennetaan verkkoon. Kaupallisissa tuotteissa tämä ominaisuus mahdollistaa käytettävän rajapinnan valitsemisen käyttökohteen tarpeiden mukaisesti. Näin voidaan varmistaa eri valmistajien tuotteiden välinen yhteentoimivuus.

Harjoitustyössä käytetään simulointiympäristön kuvan 1. mukaista erillistä osaa. Laitteiston vasemman puoleiseen Neuron-piiriin kehitetään verkkoon kirjoittava sovellus, joka suorittaa prosentteina ilmoitettavaa tasosäätöä standardin verkkomuuttujan ”SNVT_switch” tai ”SNVT_lev_percent” välityksellä. Säädön kohteena oleva suure voi olla esim. sijainti, nopeus tai intensiteetti. Verkkoon kirjoitettavaa prosenttilukua muutetaan SPI-väylään kytketyillä painonapeilla.

Oikeanpuoleisen Neuron-piirin sovellus reagoi verkkoon kirjoittavan solmun suorittamaan tasosäätöön samantyyppisten sisääntuloverkkomuuttujiensa kautta. Vastaanotettu tasosäätö esitetään promillelukemana SPI-väylään kytketyllä LCD-näytöllä.

Harjoitustyön sujuvan etenemisen varmistamiseksi painonappeja ohjaava when-lause sekä näyttöä ohjaava funktio annetaan nyt valmiina.



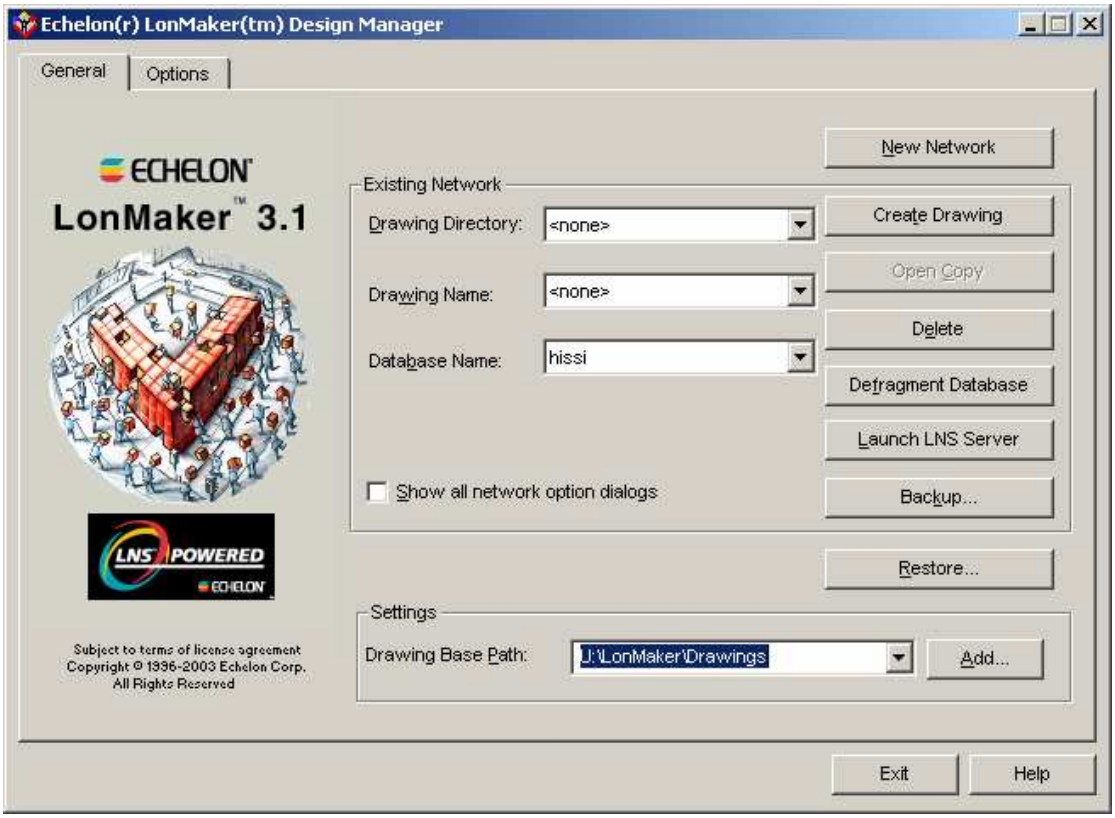
Kuva 1. Harjoitustyössä käytettävä laitteisto.

Kertaa työkurssin valmistelleen diplomityön lukua 4., ”Neuron C -ohjelmointi”. Erityisesti tutustu sovelluskerroksen verkkomuuttujaan, toiminnalliseen lohkoon sekä suuntaajafunktioon.

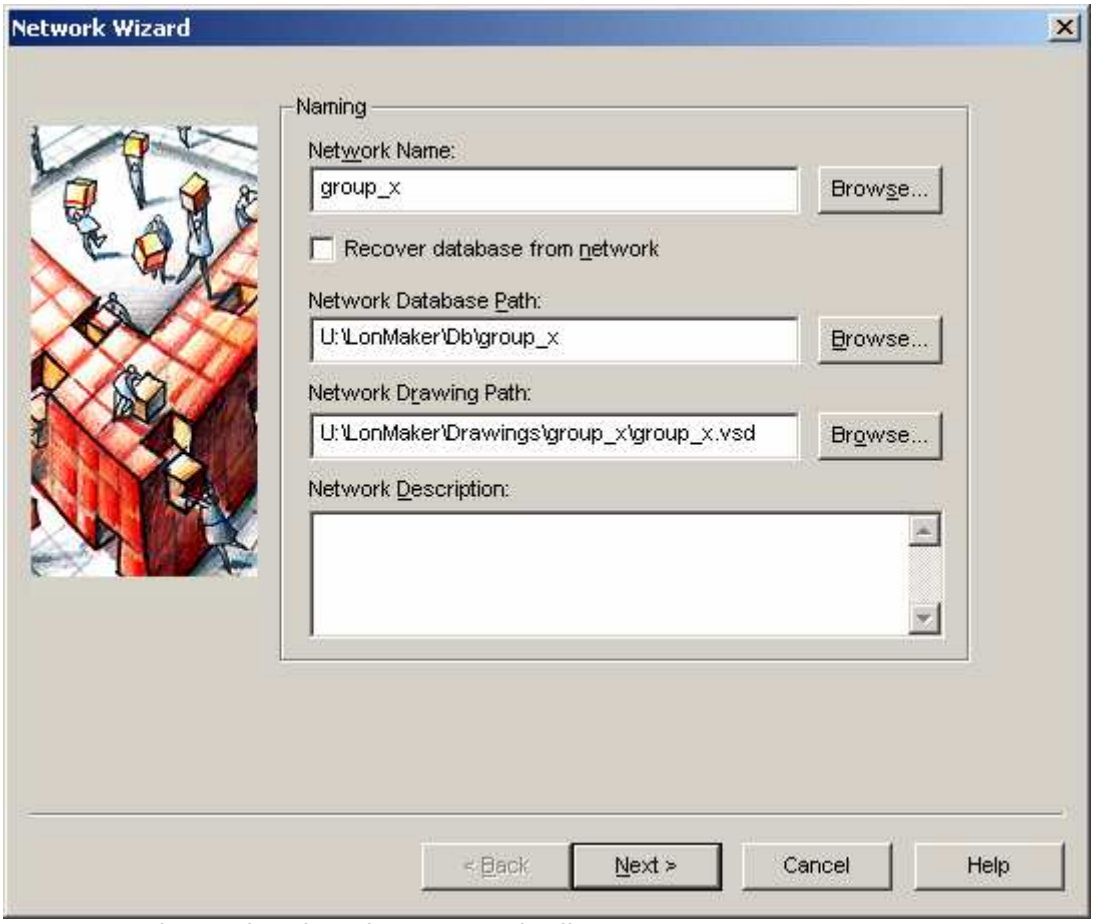
L5.1 Verkon luominen LonMakerilla

Ennen kuin sovellusohjelmien kehitystä aloitetaan NodeBuilderilla, luodaan sovellusten toteuttamille solmuille verkko LonMakeriin. Näin NodeBuilderilla käännetyt binäärit assosioituvat valmiiksi oikeaan osoitteeseen LonMakerissa.

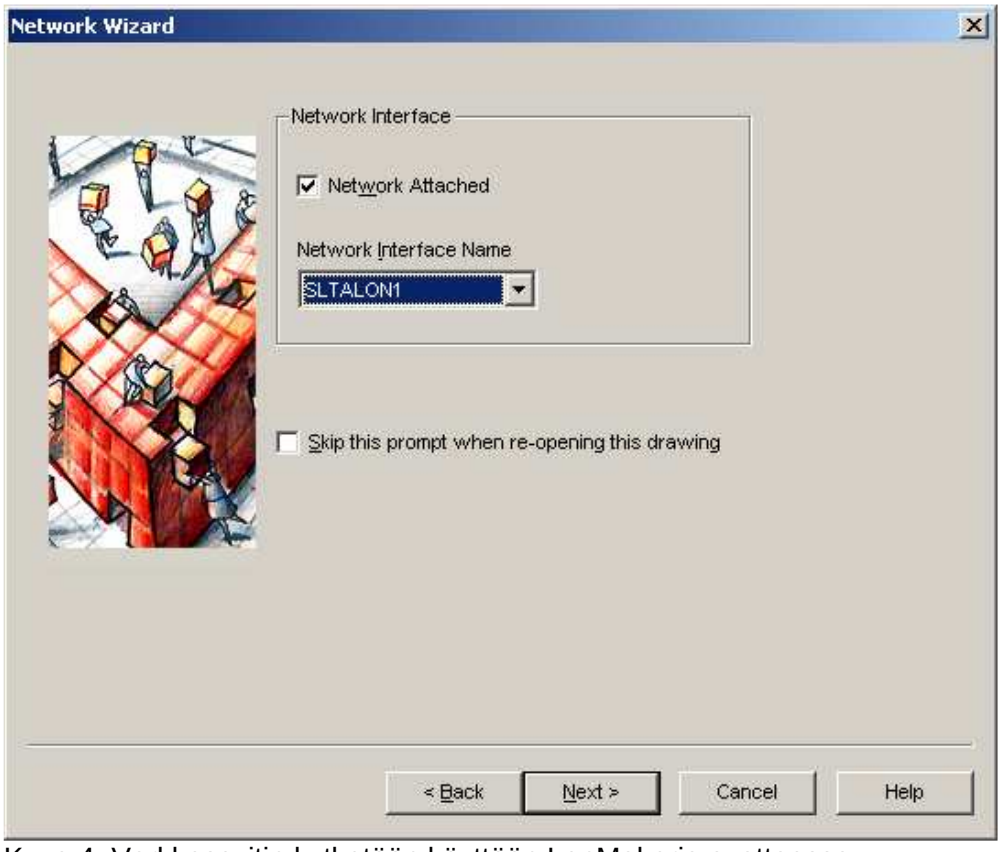
Luo LonMakeria varten omaan kotihakemistoosi verkonhallinnan tietokannalle hakemisto ”\LonMaker\Db” sekä verkon piirustuksille hakemisto ”\LonMaker\Drawings”. Käynnistä LonMaker ja aseta kuvan 2. esittämän käynnistysikkunan kenttään ”Drawing Base Path” luomasi piirustusten hakemisto. Paina nappia ”New Network”, jolloin Network Wizard aukeaa kuvan 3. mukaisesti. Tarkasta että hakemistopolut vastaavat luomiasi alihakemistoja ja aseta uuden verkon nimeksi ”group_x”, jossa x on ryhmänne numero.



Kuva 2. LonMakerin käynnistysikkuna, kun verkkoa ei ole vielä luotu.

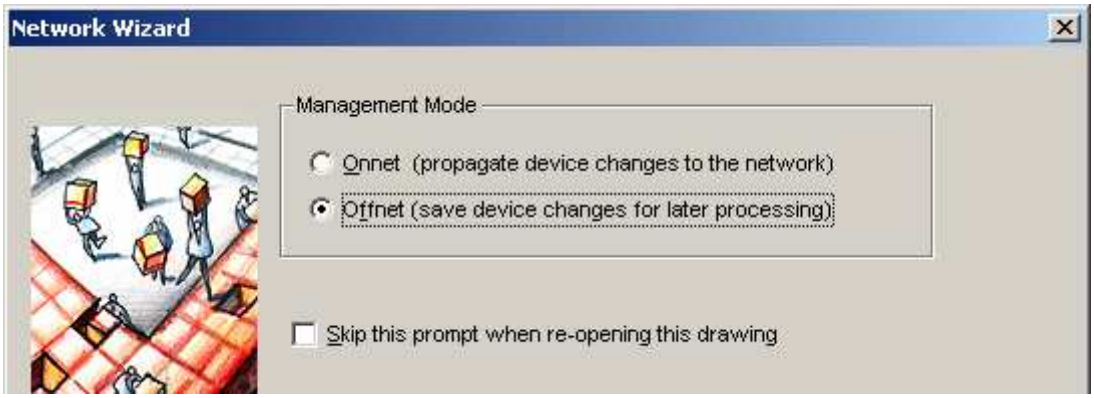


Kuva 3. Uuden verkon luominen LonMakeriin.

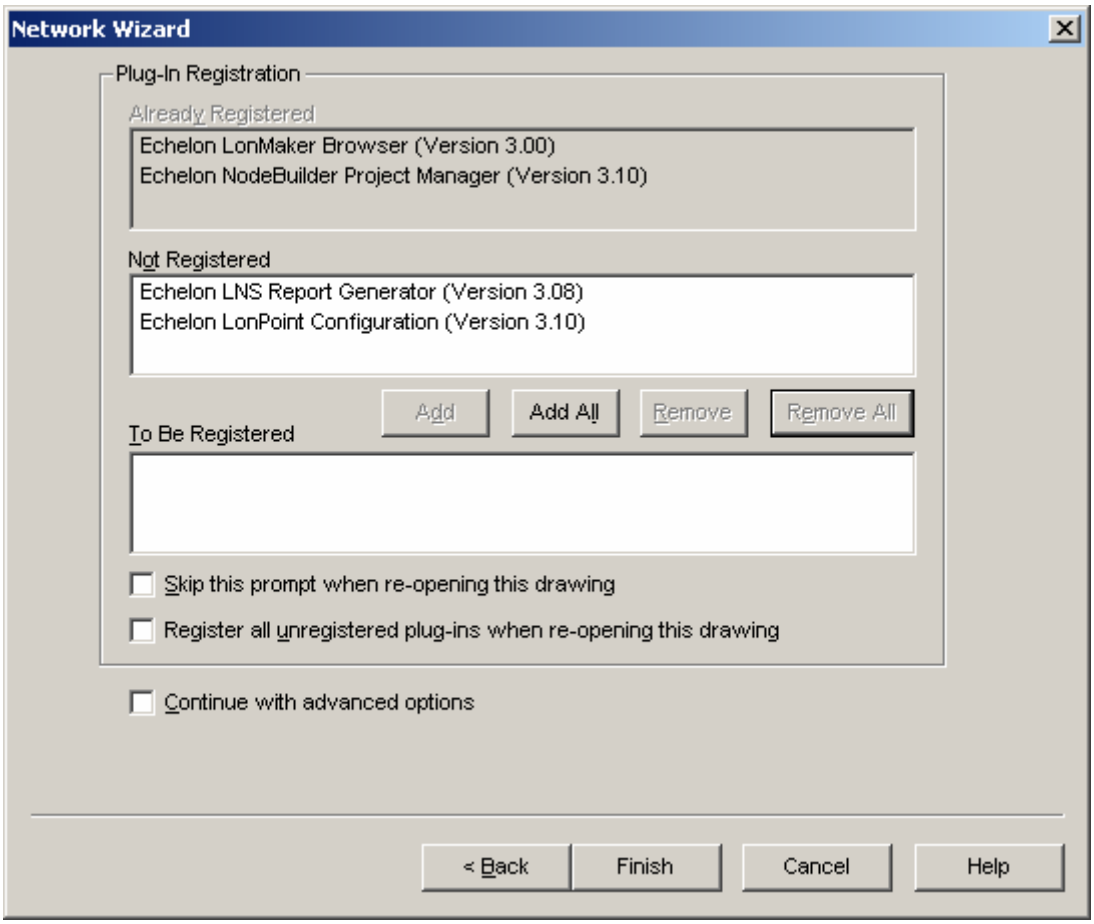


Kuva 4. Verkkosovitin kytketään käyttöön LonMakeria avattaessa.

Verkkosovitin asetetaan käyttöön kuvan 4. mukaisesti, ja verkonhallintatoimenpiteiden päivittyminen kytketään pois kuvan 5. mukaisesti. Turvallisuussyistä päivittyminen kytketään päälle vasta juuri ennen valmiin sovellusohjelman siirtoa Neuron-piirille. Poista lopuksi tarpeettomat lisäohjelmat painamalla ”Remove All”-nappia kuvan 6. mukaisesti. Talleta aukeava Microsoft Visio -piirros, tiedoston nimi ja kohde määriteltiin jo kuvassa 3. LonMaker luo nyt tietokannan uudelle verkolle tehtyjen asetusten mukaisesti. Siirry NodeBuilderiin luomaan verkossa käytettävää Neuron-piirin sovellusohjelmaa.



Kuva 5. Verkonhallinnan toimenpiteet eivät vielä etene fyysiseen Lon-verkkoon.



Kuva 6. Poista tarpeettomat lisäohjelmat painamalla ”Remove All”-nappia.

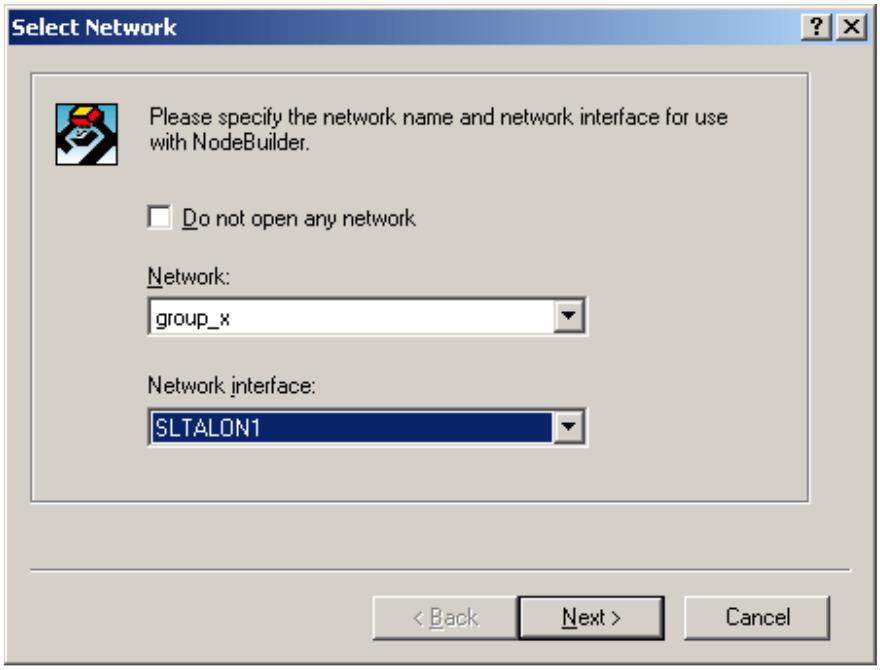
L5.2 Verkkoon kirjoittavan solmun luominen

Tässä harjoitustyössä tuotetaan kaksi sovellusohjelmaa, joista ensimmäinen kirjoittaa verkkoon vaihtoehtoisesti joko verkkomuuttujan ”SNVT_switch” tai ”SNVT_lev_percent” välityksellä. Valmiissa sovelluksessa haluttu verkkomuuttuja voidaan valita käyttöön kytkemällä sen toteuttava toiminnallinen lohko päälle.

Sovellusohjelmaa varten luodaan projekti Nodebuilderiin, generoidaan sovellukselle verkkoliikenteen rajapinta Code Wizardilla sekä täydennetään automaattisesti luotu ohjelman runko toimivaksi sovellusohjelmaksi.

L5.2.1 Projektin luominen

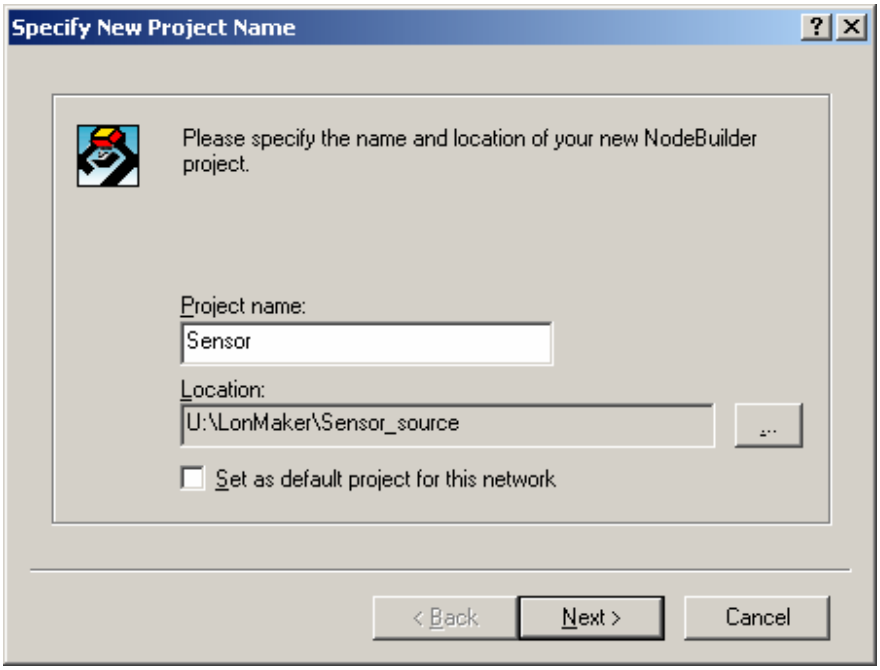
Luo NodeBuilderiin uusi projekti valikosta File, Create Project. Valitse avautuvaan ikkunaan verkkosovitin sekä edellisessä kohdassa luotu LonMaker-verkko kuvan 7. mukaisesti.



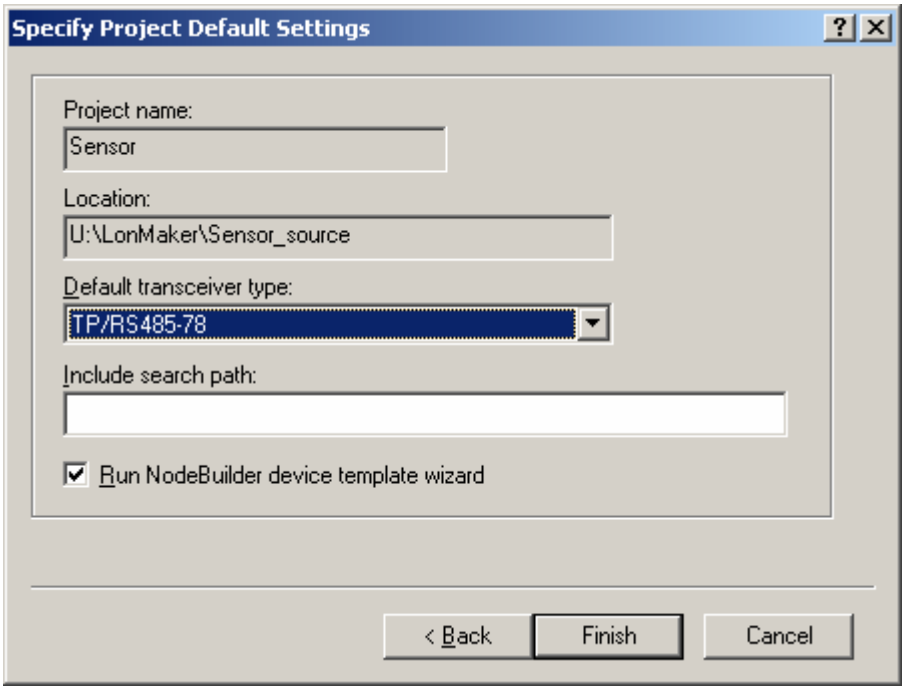
Kuva 7. Valitse LonMaker-verkko sekä verkkosovitin uudelle projektille.

Luo kirjoittavan solmun lähdekoodeja varten omaan kotihakemistoosi polku ”\LonMaker\Sensor_source” sekä käännettyjä binääritiedostoja varten hakemistopolku ”\Lonmaker\Sensor_output”.

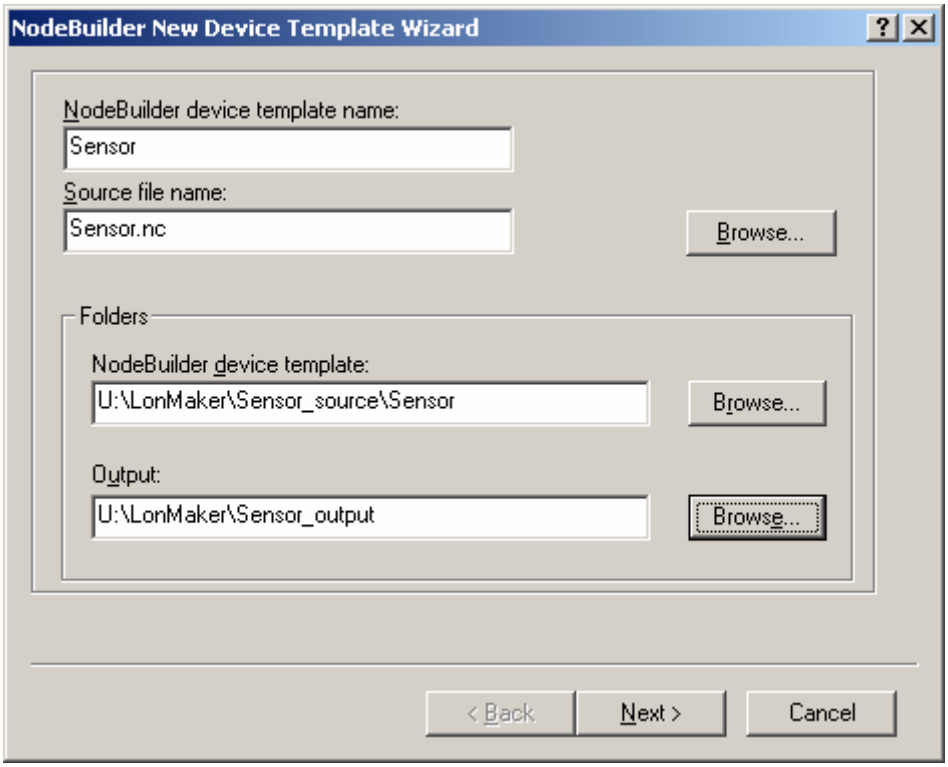
Projektia luodessasi anna sen nimeksi ”Sensor” ja aseta kotihakemistosi polku ”\LonMaker\Sensor_source” lähdekoodien hakemistoksi kuvan 8. mukaisesti.



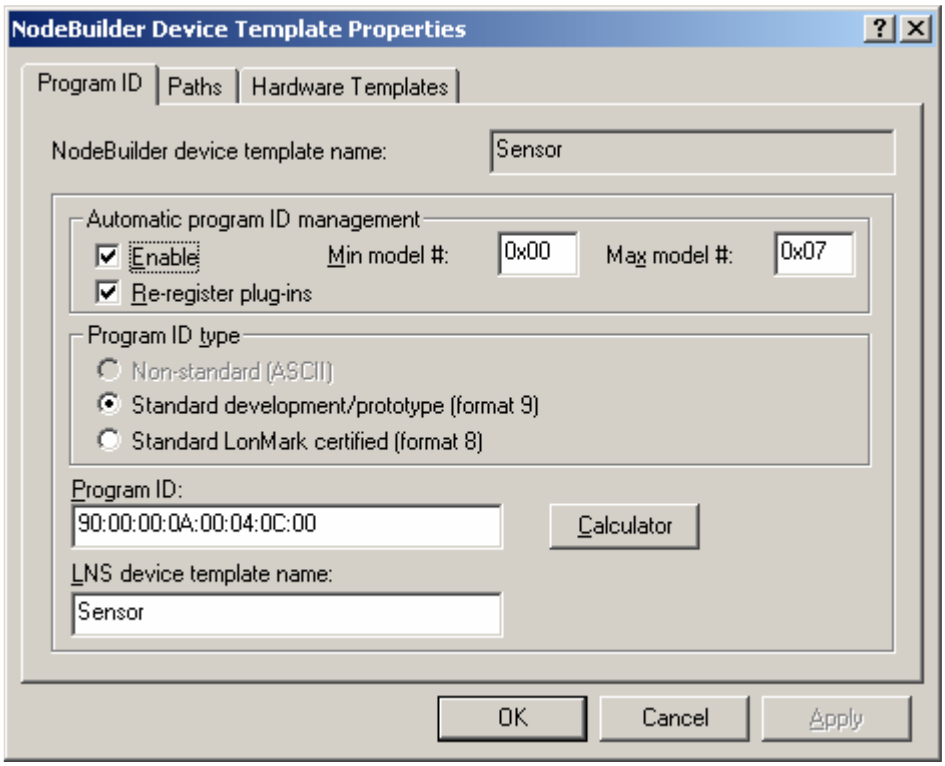
Kuva 8. Projektin nimen ja lähdekoodihakemiston asettaminen.



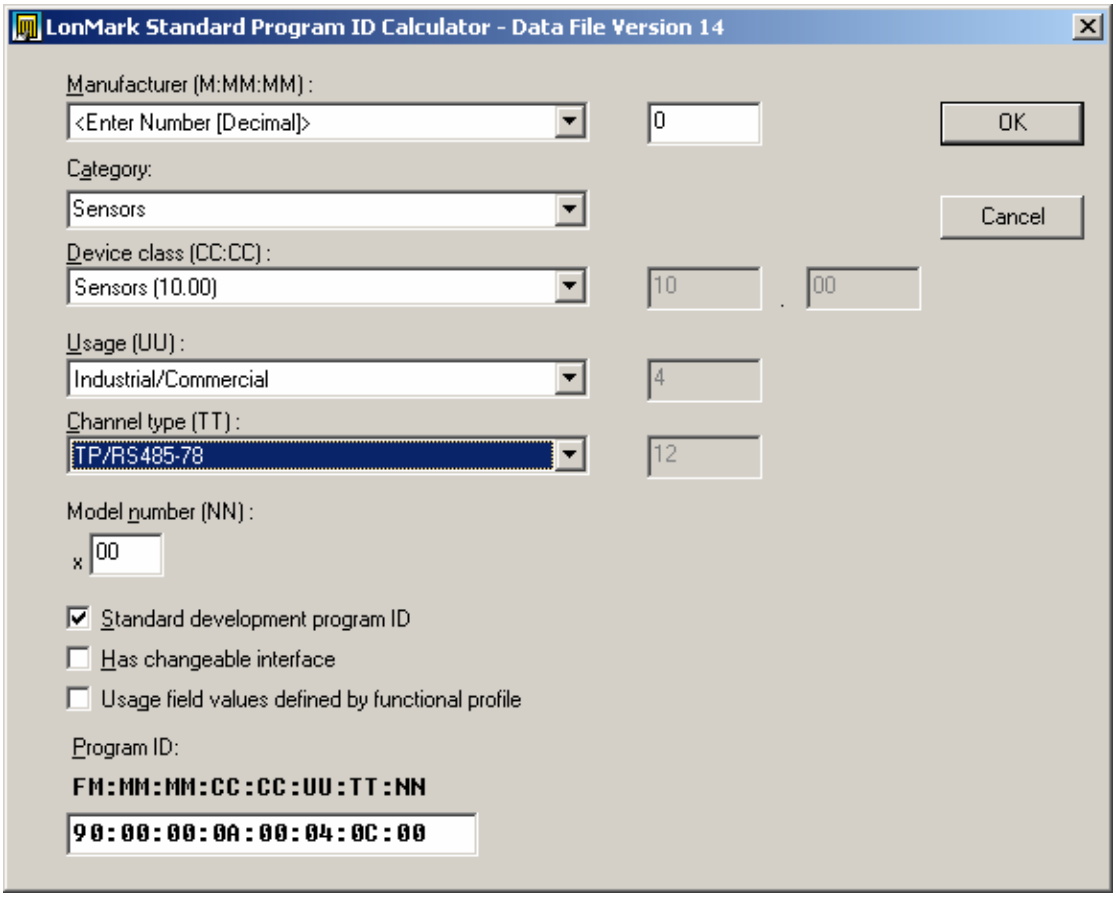
Kuva 9. Valitse lähetin-vastaanottimeksi TP/RS485-78.



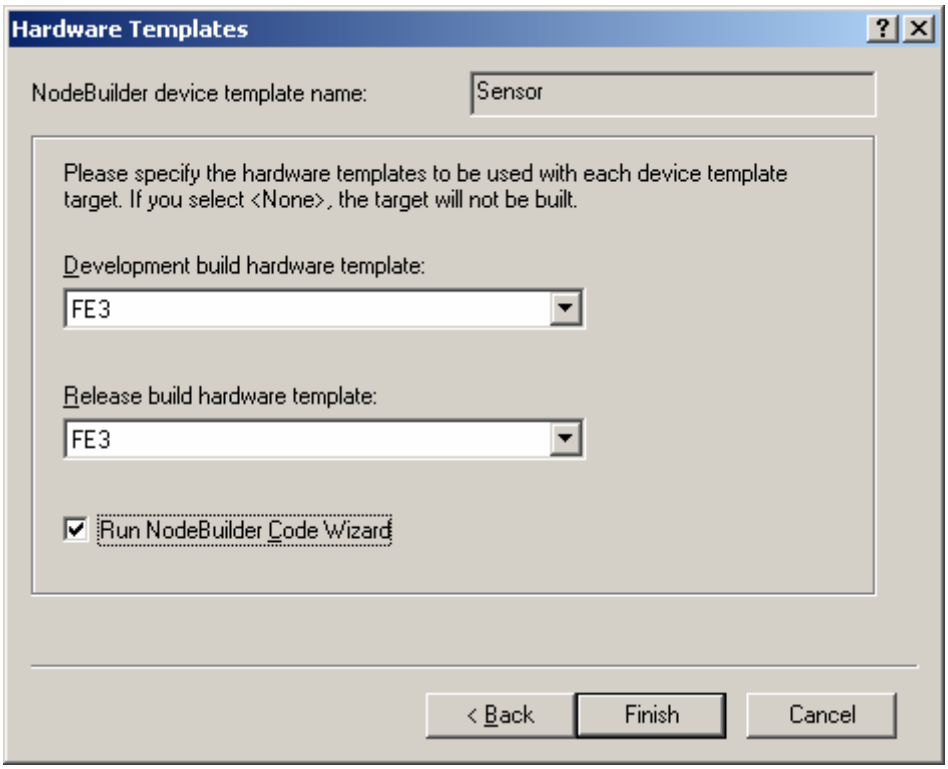
Kuva 10. Kirjoita laitemallin nimeksi ”Sensor”. Aseta käännetyt tiedostot valmistumaan kotihakemistosi alihakemistoon ”\LonMaker\Sensor_output”.



Kuva 11. Käytä tunnuksen muodostamiseen laskuria kuvan 12. mukaisesti.



Kuva 12. Aseta ohjelman tunnuksen laskuriin lähetin-vastaanottimen tyypiksi TP/RS485-78. Valitse laitetyypin tunnukseksi “Sensors (10.00)”.

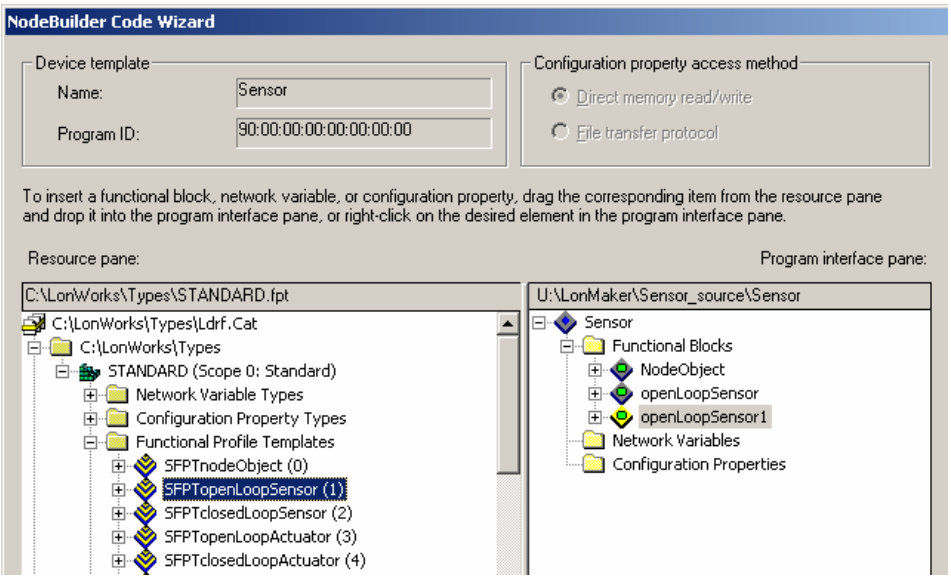


Kuva 13. Valitse laitteiston mallitiedostot sekä aseta Code Wizard ajettavaksi.

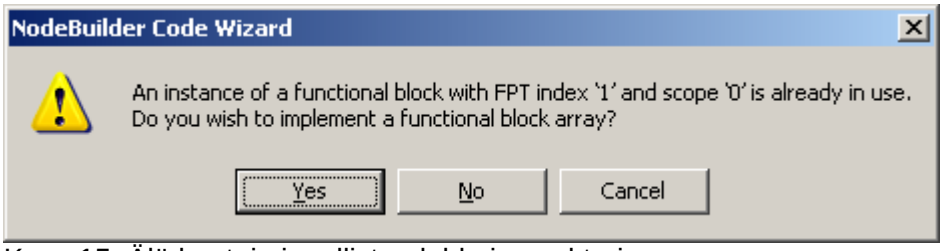
L5.2.2 Standardin verkkoliikenteen rajapinnan luonti

Sovellusohjelmalle generoidaan standardin verkkoliikenteen rajapinnan toteuttava ohjelmakoodi automaattisesti. Tämä koodi toimii muun sovellusohjelman kirjoittamisen runkona, johon haluttu toiminnallisuus täydennetään.

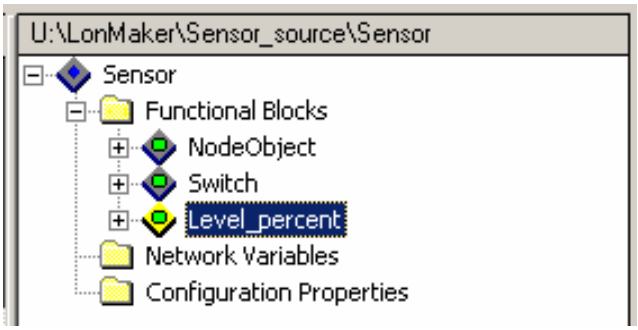
NodeBuilder-ohjelmisto sisältää resurssitiedostojen kirjaston standardien rajapinnan komponenttien luomiseksi. Kuvassa 14. näkyvässä Code Wizardin vasemmassa ikkunassa voidaan selata tarjolla olevia komponentteja. Oikean puoleinen ikkuna esittää generoinnin tuloksena syntyvän rajapinnan komponentit. Vedä hiirellä komponentti ”SFPTopenLoopSensor” oikean puoleiseen ikkunaan kahteen kertaan. Jälkimmäistä komponenttia luotaessa Code Wizard kysyy kuvan 15. tavoin, luodaanko toiminnallisten lohkojen vektori. Vastaa kieltävästi, jotta luotavat lohkot ovat toisistaan riippumattomia.



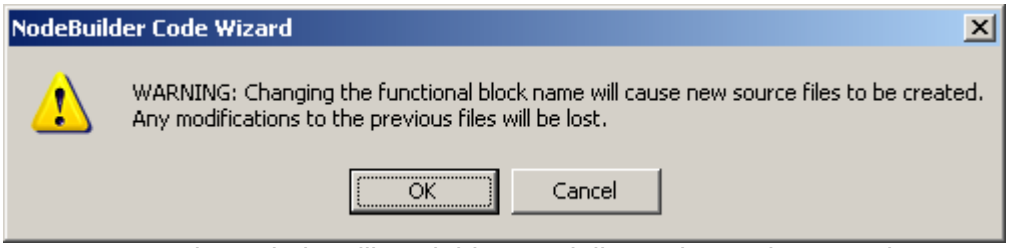
Kuva 14. Standardin verkkoliikenteen rajapinnan luonti Code Wizardilla.



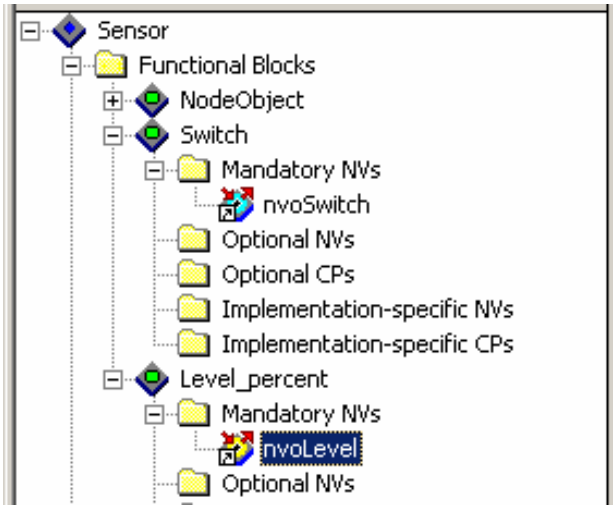
Kuva 15. Älä luo toiminnallisten lohkojen vektoria.



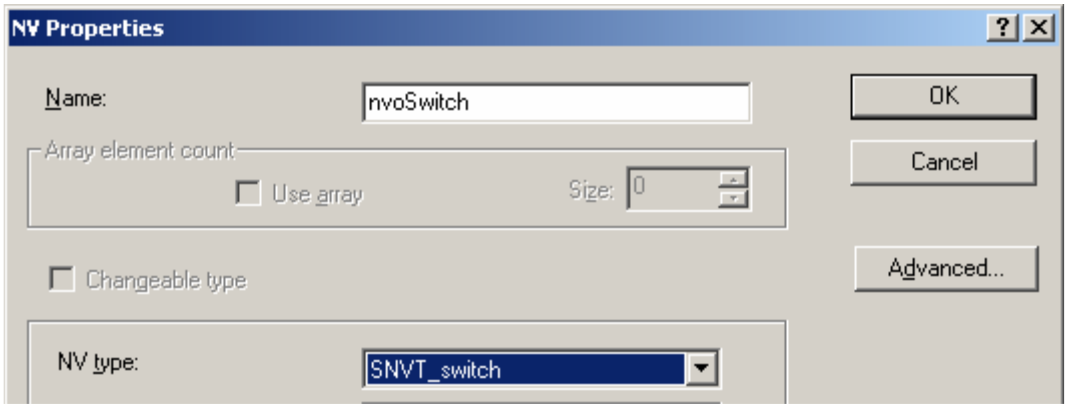
Kuva 16. Nimeä toiminnalliset lohkot uudelleen hiiren oikean napin valikosta.



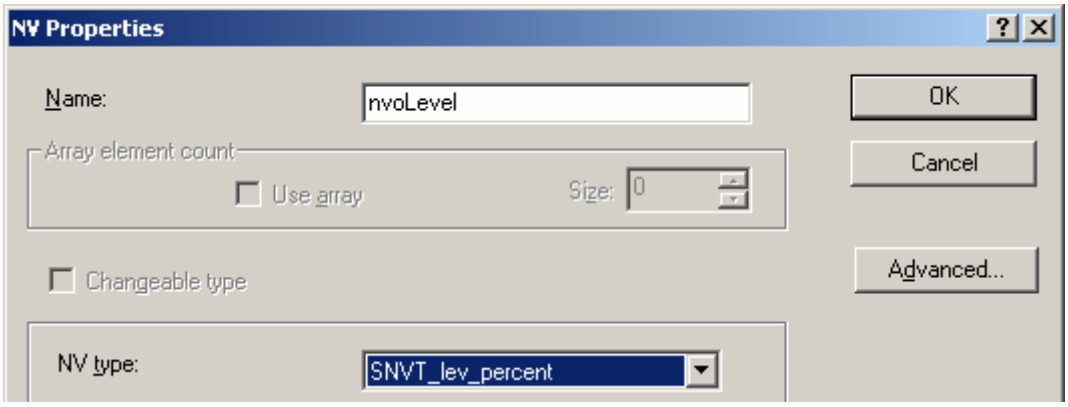
Kuva 17. Hyväksy toiminnallisen lohkon uudelleen nimeämisen varoitus.



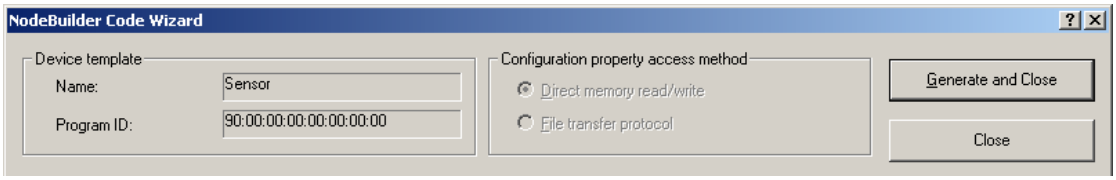
Kuva 18. Nimeä toiminnallisten lohkojen pakolliset verkkomuuttujat uudelleen.



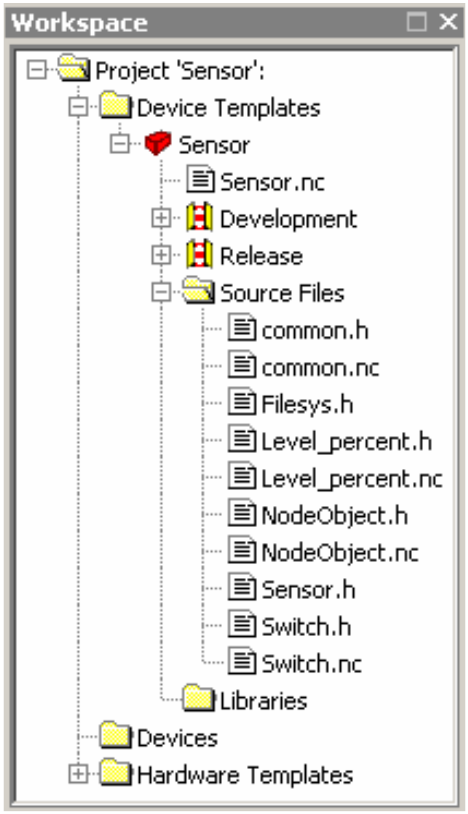
Kuva 19. Aseta muuttujan nvoSwitch tyypiksi ”SNVT_switch”.



Kuva 20. Aseta muuttujan nvoLevel tyypiksi ”SNVT_lev_percent”.



Kuva 21. Paina Code Wizardin ”Generate and Close”-nappia.

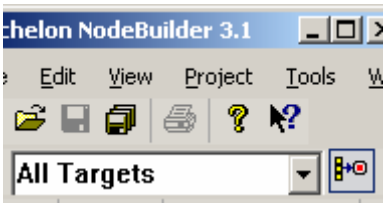


Kuva 22. NodeBuilderin Workspace-ikkuna koodin generoimisen jälkeen.

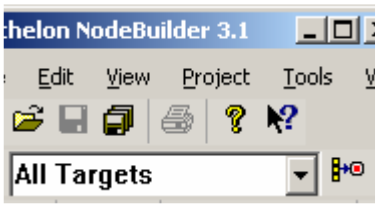
Code Wizard luo NodeBuilderiin kuvan 22. mukaiset tiedostot, jotka yhdessä toteuttavat määritellyn verkkoliikenteen rajapinnan. Näiden tiedostojen sisältämä ohjelmakoodi toimii tuotetun sovellusohjelman runkona, ja sovellusohjelmalle rakennetaan haluttu toiminnallisuus näihin tiedostoihin ohjelmakoodia täydentämällä. Tiedostot saadaan avatuksi editori-ikkunaan niitä kaksoisnapsauttamalla.

Sensor.nc toimii ohjelman päätiedostona, johon kaikki muut tiedostot on linkitetty include-lauseilla. Code Wizardilla ohjelmaan luotiin kaksi toiminnallista lohkoa, ”Switch” ja ”Level_percent”. Lisäksi automaattisesti generoitu rakenne sisältää NodeObject-lohkon, joka mahdollistaa muiden toiminnallisten lohkojen hallinnan valmiissa sovelluksessa. Common-tiedostot sisältävät muun ohjelmarungon käyttämien funktioiden määrittelyt. Filesys.h sisältää asetustiedostojen toteuttamiseen tarvittavia määrittelyjä.

Kytke pois päältä kuvassa 23. päällä oleva automaattinen käännetyn binääritiedoston kohdepiirille siirto. Näin keskeneräistä sovellusohjelmaa voidaan turvallisesti kääntää NodeBuilderissa ja siirtää vasta lopullinen versio binäärikoodista kohdepiiriin. Siirto suoritetaan tällöin LonMaker-verkonhallintaohjelmalla.



Kuva 23. Kytke pois automaattinen käännetyn binäärin kohdepiirille siirto.



Kuva 24. Automaattinen siirto on pois päältä, kun toiminnon napissa ei ole kehystä.

Älä editoi harmaana näkyvää koodia, sillä se on Code Wizardin aktiivisesti muokkaama osa ohjelmakoodia. Code Wizard voidaan käynnistää uudelleen koska tahansa sovellusohjelman kehityksen aikana ja toiminnallisten lohkojen toteuttamiin verkon rajapinnan komponentteihin voidaan tehdä muutoksia. Code Wizard käynnistyy napsauttamalla hiiren oikealla näppäimellä Workspace-ikkunassa näkyvää laitemallia (Device Template) ja valitsemalla avautuvasta valikosta ”Code Wizard”.

L5.2.3 Ohjelmarungon täydennys toimivaksi sovellusohjelmaksi

Kehitettävä sovellusohjelma vastaanottaa painonapeilla annettavia syötteitä, joiden ohjaamana se kirjoittaa ulostuloverkkomuuttujaan prosentteina ilmaistavaa tasosäätöarvoa. Lohko “Switch” toteuttaa SNVT_switch -tyyppisen verkkomuuttujan, jonka arvoa muutetaan nappeja painettaessa seuraavasti:

Nappi 1: +5 %
Nappi 2: +0,5 %
Nappi 3: -0,5 %
Nappi 4: -5 %,
Open : 100 %
Close : 0 %

Alimuuttuja “state” saa arvon nolla, mikäli alimuuttuja “value” on nolla, muuten “state” on yksi. Tutustu standardimuuttujien määrittelyyn LonMarkin oppaasta, joka löytyy ohjelmointi-PC:ltä valikosta Start - Programs - Echelon NodeBuilder - LonMark SNVT and SCPT Guide. Valitse ”SNVT Master List”. Nämä verkkomuuttujien standardimäärittelyt mahdollistavat eri valmistajien tuotteiden välisen yhteentoimivuuden.

Lohko “Level_percent” toteuttaa SNVT_lev_percent -tyyppisen verkkomuuttujan, jonka arvoa muutetaan nappeja painettaessa seuraavasti:

Nappi 1: -1 %
Nappi 2: +10 %
Nappi 3: +1 %
Nappi 4: +0,1 %,
Open: 100 %
Close: 0 %

Tasosäätöarvoa ylläpidetään laskurissa, jonka arvoa muutetaan nappia painettaessa. Koska lohkot reagoivat nappien painalluksiin eri tavalla, voidaan molempaan lohkoon sijoittaa oma laskuri. Tavoitteena on, että ainoastaan yhtä lohkoa voidaan käyttää kerrallaan, eli lohkon ”Level_percent” verkkomuuttuja ei päivity mikäli myös lohko “Switch” on päälle kytkettynä. Lähetetyt tasosäätöarvot tulee myös rajoittaa nollan ja 100 prosentin välille.

Sensor.h on sovellusohjelman päätiedoston otsikkotiedosto, joka sisältää globaaleja tyyppi- ja makromäärittelyjä. Sinne lisätään niiden ajastimien ja I/O-objektien määrittelyt, joiden tulee olla aina käytössä ja näkyä kaikkialle sovellusohjelmassa. Avaa tiedosto “pohja_sensor.txt” ohjelmointi-PC:n hakemistosta ”D:\Mallit” ja kopioi sieltä tarvittavat ajastin-, I/O- ja funktion prototyypin määrittelyt. Lisää otsikkotiedostoon omat #define-määrittelysi esim. käyttämällesi director-funktion komennolle.

Sensor.nc on sovellusohjelman päätiedosto. Se sisältää globaalit when-lauseet, joista annetaan reagointikomentoja halutuille toiminnallisille lohkoille niiden suuntaajafunktioita kutsumalla. Reagointi järjestelmätason tapahtumiin, esim. resetointiin, tapahtuu näiden when-lauseiden kautta. Alusta päätiedoston when(reset)-osiossa globaalit ajastimet ja muuttujat. Kopioi mallitiedoston sisältämä painonappeja ohjaava when-lause ja sen käyttämä to_decimal( )-funktio. Lisää when-lauseen ohjaamaan koodiin director-funktioiden kutsut, joilla tieto painetuista napeista siirretään ulostuloverkkomuuttujat sisältäville lohkoille. Suorita nämä kutsut vain tarvittaessa, mikäli kyseiset lohkot ovat asetettuina käyttöön. Tämä voidaan tarkastaa funktiokutsulla ”fblockNormalNotLockedOut(lohkon_nimi::global_index)”, joka palauttaa arvon TRUE, mikäli lohko on aktiivisena.

Switch.h sisältää lohkon ”Switch” määrittelyt. Sinne sijoitetaan tyyppi-, I/O- ja makromäärittelyt, joita käytetään vain tässä lohkossa. Määrittele sinne lohkon suorittaman tasosäädön tarvitsema laskuri.

Switch.nc sisältää lohkon “Switch” toiminnallisuuden toteuttavan ohjelmakoodin, joka voi sisältää lohkokohtaisia ajastin- ja I/O-toimintoja. Alusta lohkokohtaiset muuttujat, esim. luomasi laskuri, lohkon suuntaajafunktion FBC_WHEN_RESET-osiossa. Luo lohkon suuntaajafunktioon oma ”else if”-osiosi, joka päivittää lohkon tasosäätöarvoa ylläpitävää laskuria ja päivittää skaalatut arvot ulostuloverkkomuuttujaan.

Luo lohkoon “Level_percent” vastaava toiminnallisuus, minkä teit juuri lohkoon ”Switch”. Ulostuloverkkomuuttuja on nyt erityyppinen ja tarvitsee erilaisen skaalauksen. Ohjeet skaalaukseen löytyvät oppaasta ”LonMark SNVT and SCPT Guide”. Huomioi myös edellisellä sivulla määritelty lohkojen erilainen reagointi nappien painamiseen tasosäätöprosentin muuttamiseksi.

Tiedosto common.nc sisältää muun ohjelmarungon käyttämien funktioiden määrittelyt. Osa näistä funktioista jää käyttämättä, mikä aiheuttaa kääntäjän varoituksia. Varoituksen saa pois lisäämällä käyttämättä jäävän funktiomäärittelyn perään seuraavan kääntäjän käskyn:

```
#pragma ignore_notused funktion_nimi
```

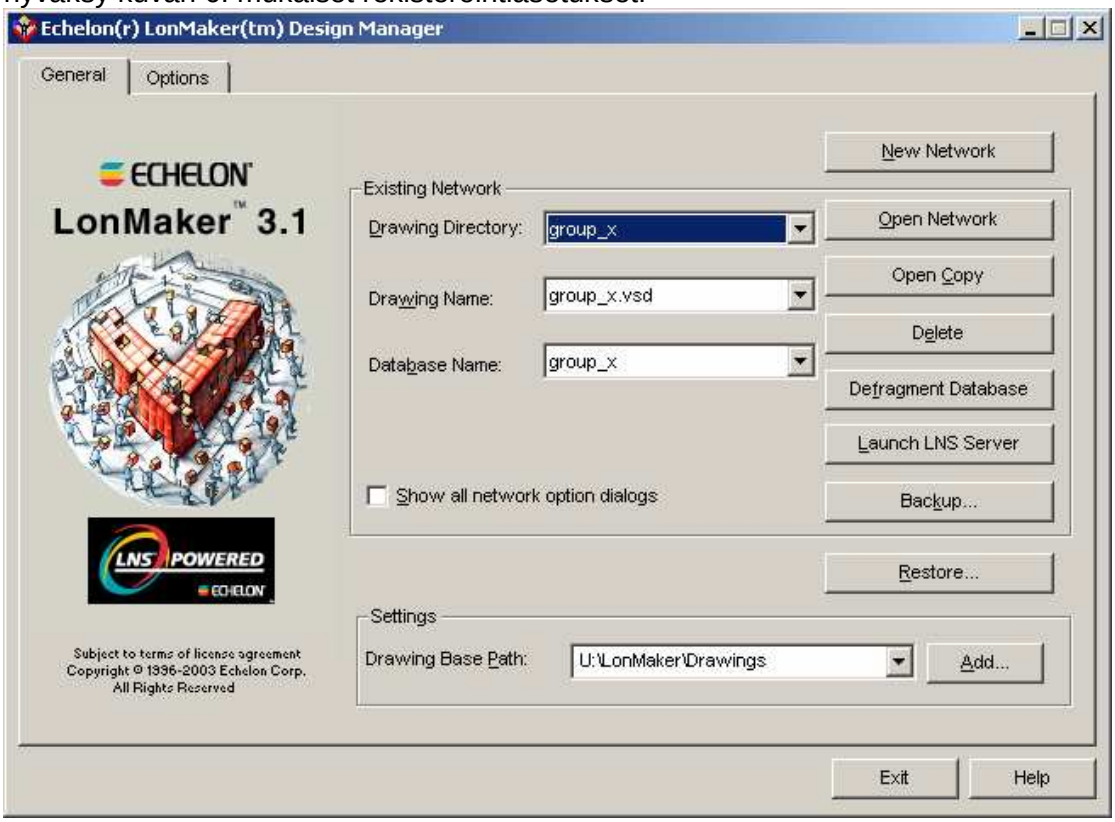
Mikäli halutaan kääntää sovellusohjelman kehitysversio, saattaa binääritiedoston koko kasvaa Neuron-piirin EEPROM-muistikapasiteettia suuremmaksi. Tällöin kääntäjä antaa virheilmoituksen ”Error: No more memory in CODE area”. Binääritiedoston kokoa voidaan pienentää muuttamalla common.nc-tiedoston sisältämät käyttämättömät funktiot kommenteiksi syntaksilla ”//”. Sovellusohjelman

julkaisuversio on kehitysversiota pienempi, koska se ei sisällä debug-toimintojen tarvitsemia ylimääräisiä merkintöjä. Julkaisuversiota onkin suositeltavaa käyttää, kunhan ”Node recovery” on kytketty laitemallin julkaisuversion asetuksiin päälle. Tarkasta, että sen asetukset vastaavat ”Simulointiympäristön käyttöohjeen” kuvaa 13. Varmistu samalla, että laitteiston mallitiedoston (Hardware Template) asetukset vastaavat ”Simulointiympäristön käyttöohjeen” kuvaa 14.

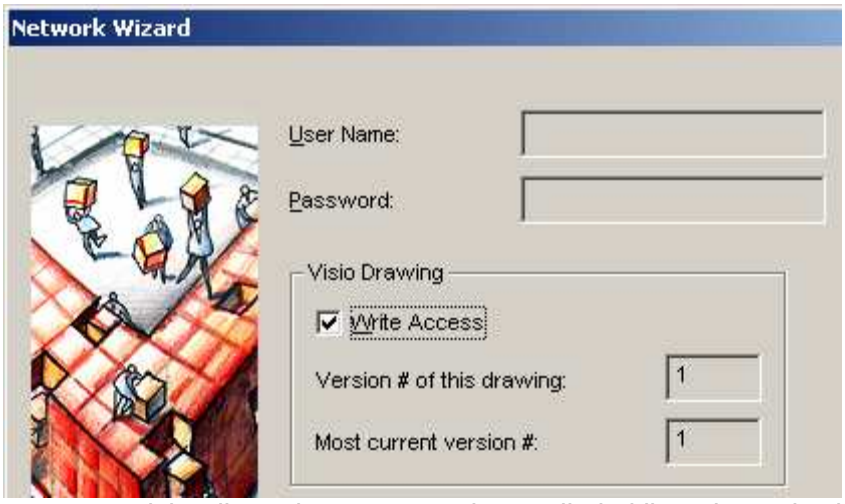
Onnistuneesti käännetty valmis sovellusohjelma siirretään Neuron-piiriin LonMaker-verkonhallintaohjelmalla. Sovellus ”Sensor” kirjoittaa verkkoon painonapeilla annettavien syötteiden mukaisesti, ja tämä sovellus siirretään simulointiympäristön kuvan 1. mukaisen erillisen osan vasemman puoleiseen Neuron-piiriin. Neuron-piirin sisältämää sovellusta voidaan testata erikseen LonMakerin Browser-toiminnolla. Sillä voidaan lukea yksittäisen sovelluksen ulostuloverkkomuuttujaansa kirjoittamia arvoja. Myös yksittäisen sovelluksen sisääntulomuuttujiin voidaan syöttää arvoja ja havainnoida syötteiden aiheuttamia reaktioita laitteessa.

L5.2.4 Valmiin sovelluksen siirto Neuron-piiriin LonMakerilla

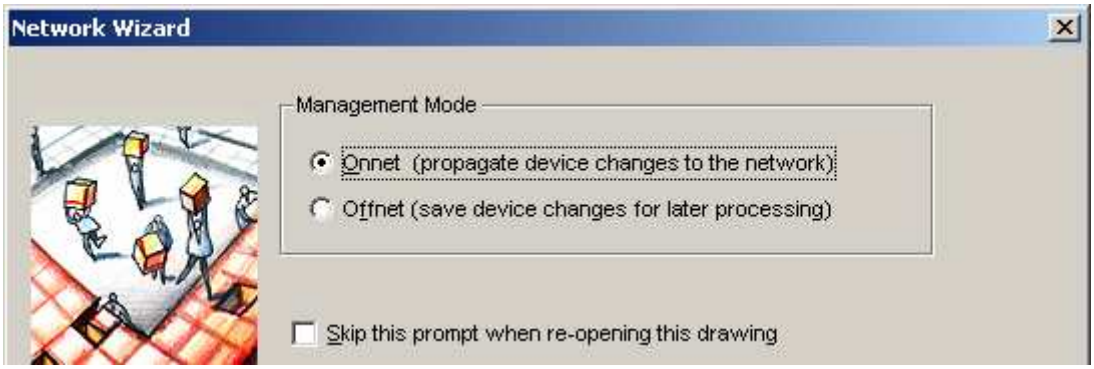
Avaa kohdassa 1.1 luomasi verkko LonMakeriin. Valitse kuvan 25. esittämän käynnistysikkunan alareunassa olevaan ”Drawing Base Path”-kenttään polku oman kotihakemistosi sisältävään LonMaker\Drawings-hakemistoon, ja valitse sitten luomasi verkon nimi ylempään ”Drawing Directory”-kenttään. Paina nappia ”Open Network”. Kytke jälleen verkkosovitin käyttöön kuvan 4. mukaisesti ja aseta verkon rakennetta kuvaava Microsoft Visio -piirustus editointitilaan kuvan 26. mukaisesti. Kytke verkonhallintatoimenpiteet etenemään verkkoon kuvan 27. mukaisesti ja hyväksy kuvan 6. mukaiset rekisteröintiasetukset.



Kuva 25. LonMakerin käynnistysikkuna, kun verkko on jo luotu.



Kuva 26. Visio-piirustuksen asettaminen editointitilaan käynnistyksen yhteydessä.



Kuva 27. Verkonhallinnan toimenpiteet kytketään etenemään Lon-verkkoon.

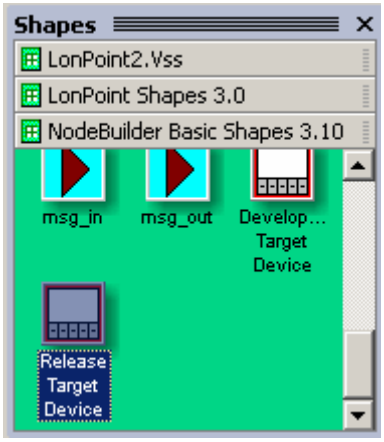
Asetusta verkonhallinnan etenemisestä fyysiseen verkkoon voidaan muuttaa valikosta LonMaker - Network Properties - Onnet/Offnet. Asetusten eteneminen on syytä katkaista, jos esim. siirrytään luomaan uutta sovellusohjelmaa NodeBuilderiin. Näin estetään keskeneräisen sovelluksen tahaton latautuminen Neuron-piiriin.

LonMaker-verkonhallintaohjelma sisältää lisenssimenetelmän, jossa ohjelmiston käyttäjä voi hallita kerralla vain rajallista määrää laitteita. Tämä mekanismi sisältää pistetilin, jonka saldon mukaisesti voidaan vielä antaa toimeksiantoja uusille laitteille. Tililtä vähennetään aina yksi piste, kun Neuron-piiri asetetaan normaaliin toimintatilaan. Vastaavasti pisteen saa takaisin, mikäli sama Neuron-piiri asetetaan myöhemmin asetuksettomaan tilaan (decommission) luotua verkon tietokantaa käyttäen. Asetuksettomassa tilassa Neuron-piirin huolto-LED vilkkuu sekunnin välein.

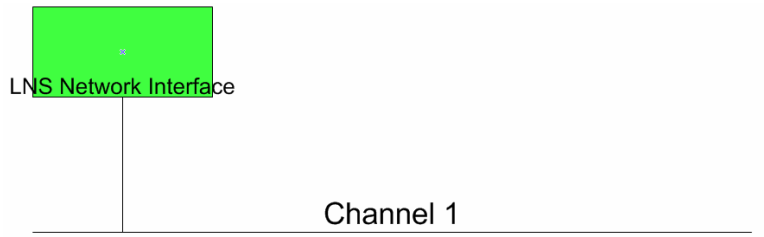
Tätä harjoitusta tehtäessä ryhmä tarkistaa ennen sovellusohjelmien siirtoa käytössä olevien toimeksiantopisteiden lukumäärän. Työvuoronsa päättyessä ryhmä luovuttaa laitteiston seuraavan ryhmän käyttöön vasta, kun varatut toimeksiantopisteet on vapautettu käyttöön. Pisteiden vapauttaminen edellyttää niiden varauksessa käytetyn tietokannan käyttöä. Pisteitä ei voida enää vapauttaa, jos seuraava ryhmä kirjoittaa Neuron-piiriin uuden sovellusohjelman ennen kuin edellisen ryhmän tekemän ohjelman sitomat pisteet on vapautettu.

Pistetilanne tarkistetaan valikosta LonMaker - LonMaker Credits Info, josta aukeaa kuvan 28. mukainen ikkuna.

Kuva 28. LonMakerin toimeksiantopisteet.



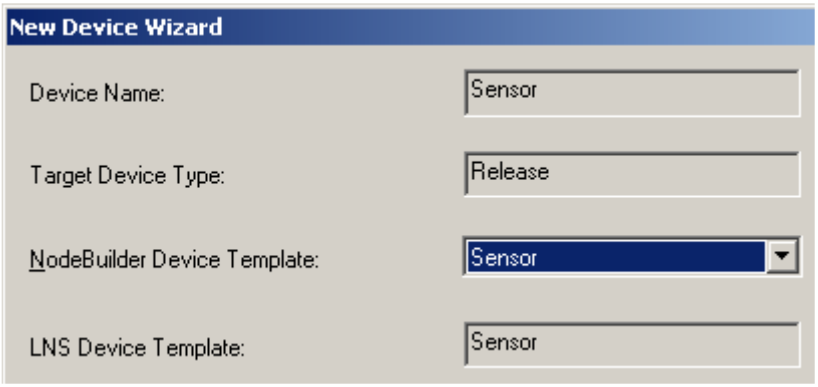
Kuva 29. Vision ”Shapes”-ikkuna sisältää kuvakkeet NodeBuilderin laitemalleille.



Kuva 30. Juuri luodun verkon fyysisen kerroksen kuvaus Visiossa.

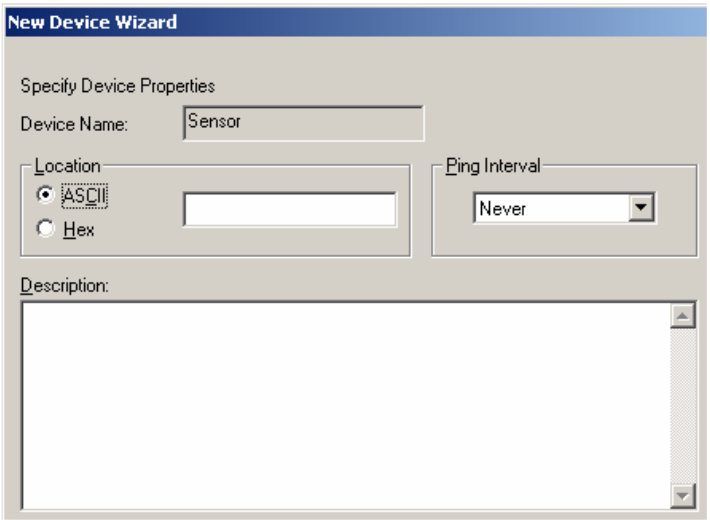
Vedä hiirellä kuvan 29. esittämä julkaisuversion laitemallin kuvake kuvan 30. esittämään verkon piirustukseen. Anna laitteelle sovelluksesi nimi, esim. ”Sensor”, ja aseta laite normaaliin toimintatilaan (Commission Device) kuvan 31. mukaisesti. Kuvan 32. esittämässä laitemallin valintaikkunassa on valmiiksi oikea laitemalli valittuna, mikäli binääritiedostoa NodeBuilderissa käännettäessä projektiasetukset ovat sisältäneet nyt käytettävän verkon.

Kuva 31. Laitteen nimeämien ja asettaminen normaalin toimintatilaan.



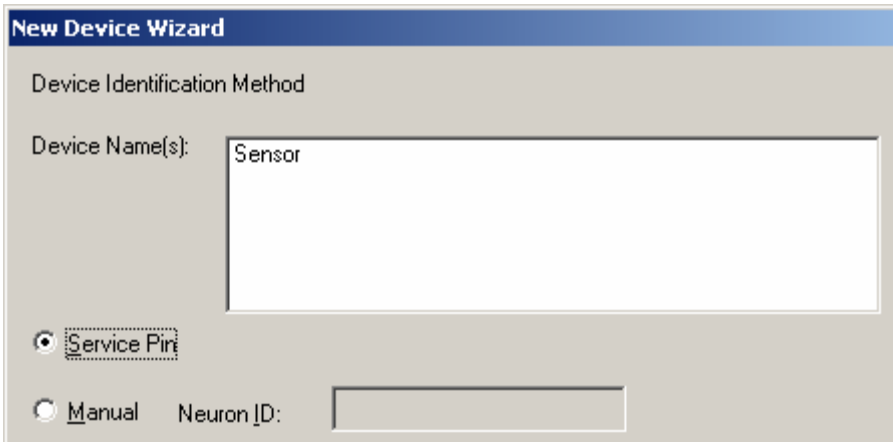
Kuva 32. Laitemallin valinta sovellusohjelman siirtoa varten.

Kuva 33. Lähetin-vastaanottimen valinta sovellusohjelman siirtoa varten.

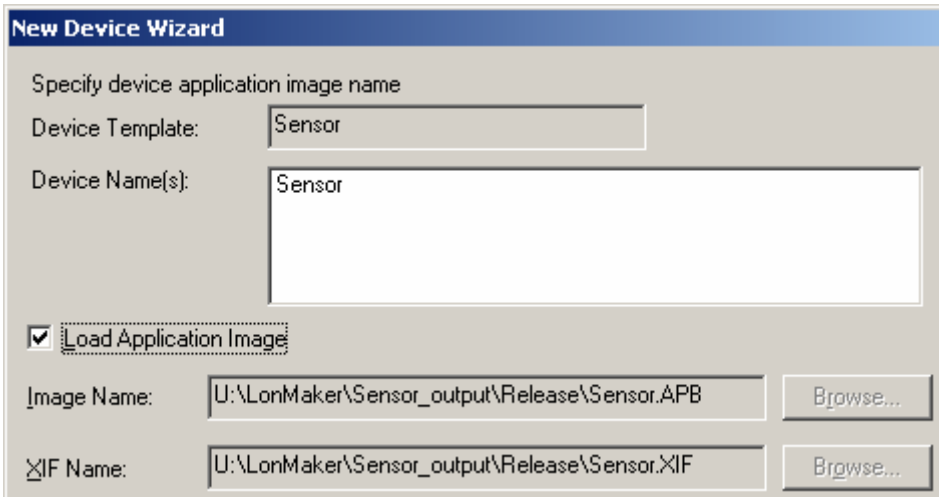


Kuva 34. Laitteen sijaintia verkossa ei tarvitse kuvailla.

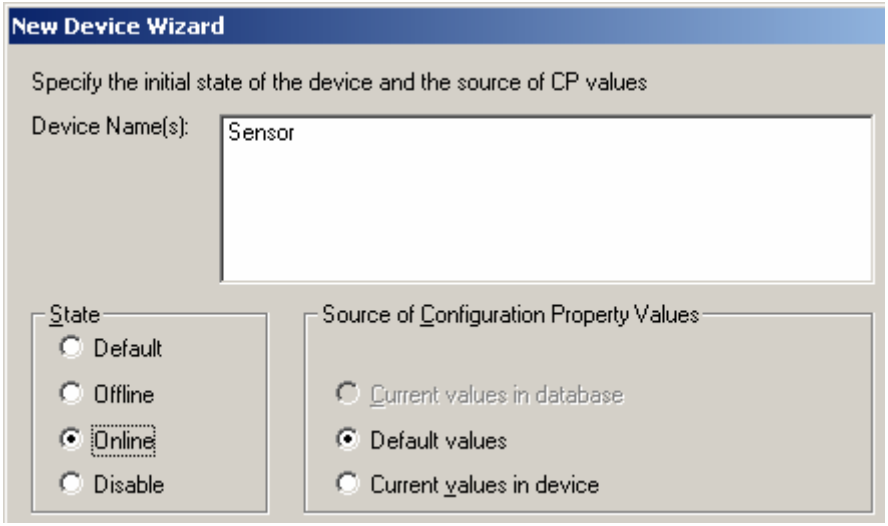
Valitse lähetin-vastaanotin kuvan 33. mukaisesti ja jätä kuvan 34. mukaisesti laitteen sijaintia ja tehtävää kuvailevat kentät tyhjiksi. Valitse Neuron-piirin tunnistus Service Pin -painalluksella kuvan 35. mukaisesti. Tarkista binäärikoodin hakemistopolku sekä merkitse sovellus ladattavaksi kuvan 36. esittämään ikkunaan. Hakemistopolku on valmiiksi oikein, mikäli binääritiedostoa NodeBuilderissa käännettäessä projektiasetukset ovat sisältäneet nyt käytettävän verkon.



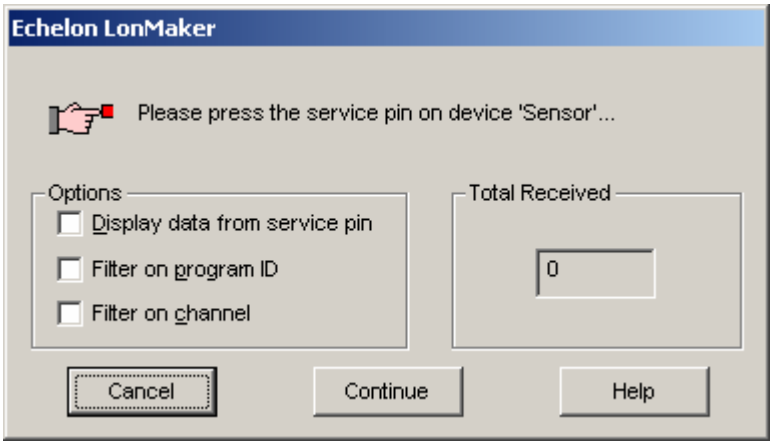
Kuva 35. Valitse Neuron-piirin tunnistus Service Pin -painalluksella.



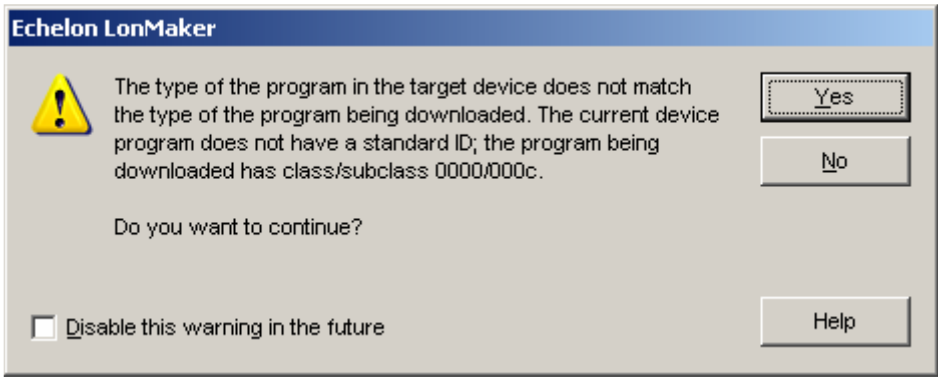
Kuva 36. Tarkista binäärikoodin hakemistopolku ja merkitse sovellus ladattavaksi.



Kuva 37. Valitse laitteen alkutilaksi ”Online”.



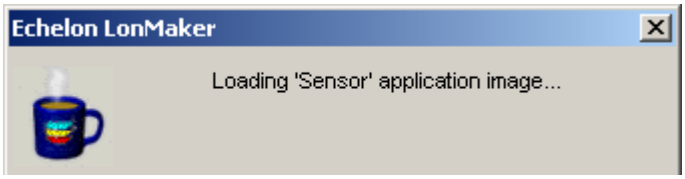
Kuva 38. Paina Neuron-piirin fyysistä Service Pin -nappia.



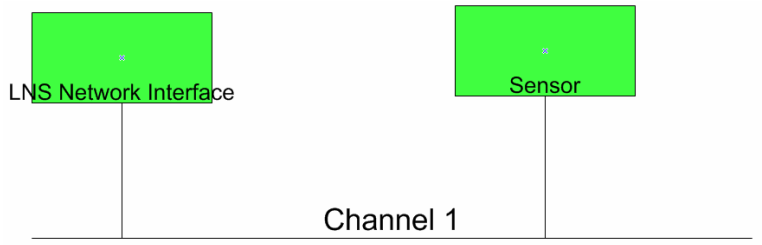
Kuva 39. Varoitus siirrettävän ja piirillä jo olevan sovelluksen erilaisista tunnuksista.

Kuvan 38. näkyessä LonMaker odottaa Service Pin -painallusta. Painalluksen jälkeen esiintyy kuvan 39. mukainen varoitus, mikäli Neuron-piiri sisältää ennestään erilaisen sovellusohjelman tunnuksen kuin sinne nyt siirrettävä sovellus. Varoituksen voi sivuuttaa, kunhan uuden sovelluksen lähetin-vastaanottimen ja Neuron-piirin tyyppiä koskevat asetukset ovat oikein, eli NodeBuilderin laitteiston mallitiedoston (Hardware Template) asetukset vastaavat ”Simulointiympäristön käyttöohjeen” kuvaa 14. Mikäli NodeBuilderin asetuksissa huomataan virhe, on binääri käännettävä uudelleen NodeBuilderissa ennen sovelluksen latausta Neuron-piiriin.

Sovellusohjelman latautuminen piiriin ilmenee kuvan 40. mukaisesta ikkunasta sekä Neuron-piirin epätasaisesti välkkyvästä huolto-LEDistä.



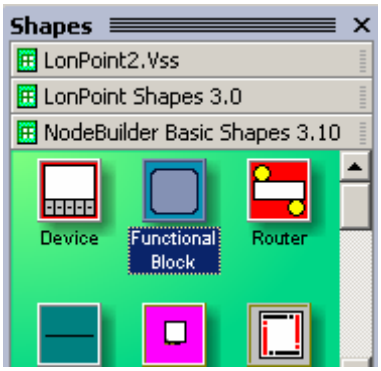
Kuva 40. Sovellusohjelma latautuu Neuron-piiriin.



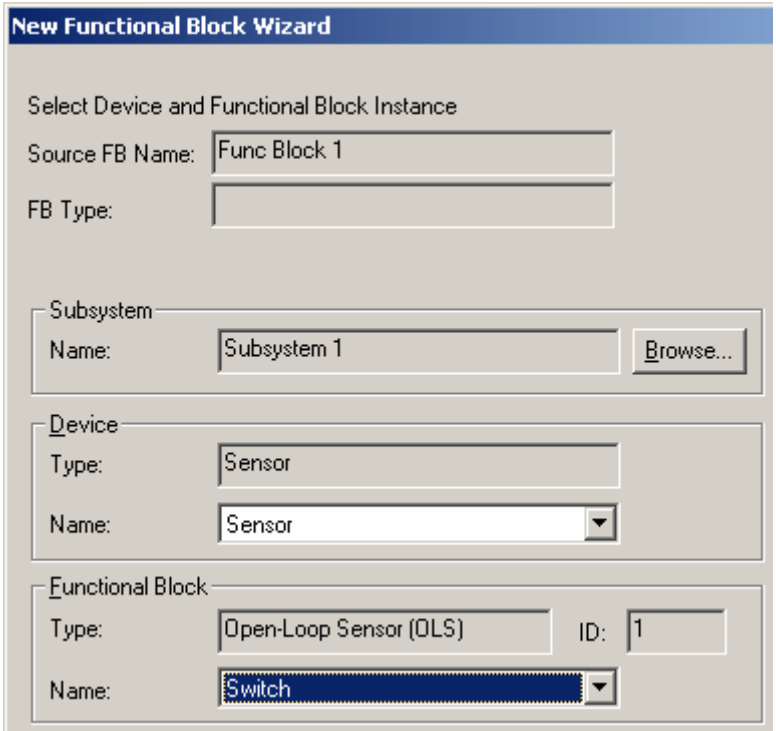
Kuva 41. Verkon fyysisen kerroksen kuvaus Visiossa laitteen lisäyksen jälkeen.

Kuva 41. esittää verkon fyysisen kerroksen kuvauksen Visiossa laitteen lisäyksen jälkeen. Vihreänä näkyvät laitteet ovat asetettuina normaaliin toimintatilaan, keltaisena näkyvälle laitteelle ei ole fyysistä vastinetta LonMakerin tietokannassa. Laitteet voivat näkyä vihreinä myös Offline-tilassa, tällöin LonMaker tukeutuu tietokantansa sisältämiin asetustietoihin. Tehdyt uudet asetukset tallentuvat tietokantaan ja päivittyvät seuraavan kerran Online-tilaan siirryttäessä. Tämän ominaisuuden ansiosta laajan verkon asetukset voidaan tehdä valmiiksi toimistossa, jolloin asetukset voidaan nopeasti siirtää fyysiseen asennuskohteeseen esim. rakennustyömaalla.

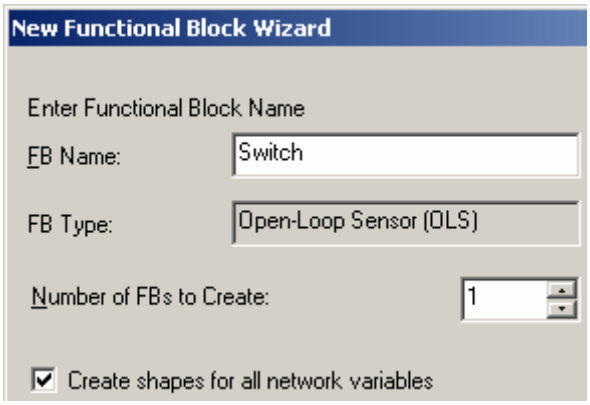
Verkon fyysisen kerroksen esitykseen voidaan nyt lisätä loogisen kerroksen esitys. Tämä tehdään vetämällä hiirellä kuvan 42. esittämä toiminnallisen lohkon kuvake kuvan 41. esittämään verkon graafiseen kuvaukseen. Molemmat itse luodut toiminnalliset lohkot, NodeObject-lohko sekä virtuaalinen lohko vedetään kukin erikseen hiirellä Shapes-ikkunasta samaa kuvaketta käyttäen. Virtuaalinen lohko sisältää ne verkkomuuttujat, jotka sijaitsevat jonkin toiminnallisen lohkon sijasta sovellusohjelman perusosassa. Harjoitustyön virtuaaliset lohkot ovat näin ollen tyhjiä.



Kuva 42. Vision ”Shapes”-ikkuna sisältää toiminnallisen lohkon kuvakkeen.

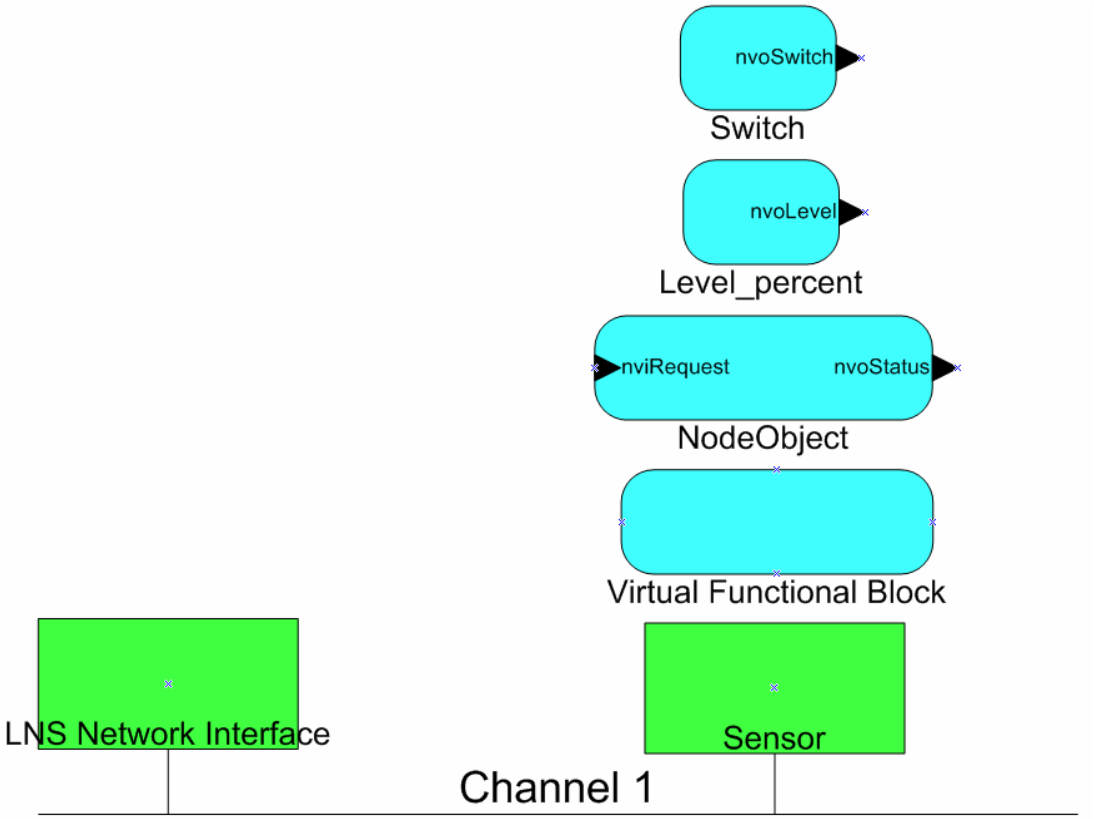


Kuva 43. Toiminnallisen lohkon graafisen esityksen luonti.



Kuva 44. Lohkon nimeä voidaan vielä vaihtaa. LUO KAIKKI HAHMOT (shapes)!

Toiminnallisen lohkon graafisen esityksen luominen vahvistetaan kuvan 43. mukaisessa ikkunassa, jossa voidaan valita mikä sovelluksen sisältämistä lohkoista piirretään. Kuvan 44. ikkunassa piirrettävä lohko voidaan vielä nimetä uudelleen. Laita rasti ruutuun, jotta kaikille lohkon sisältämille verkkomuuttujille luodaan tunnus.



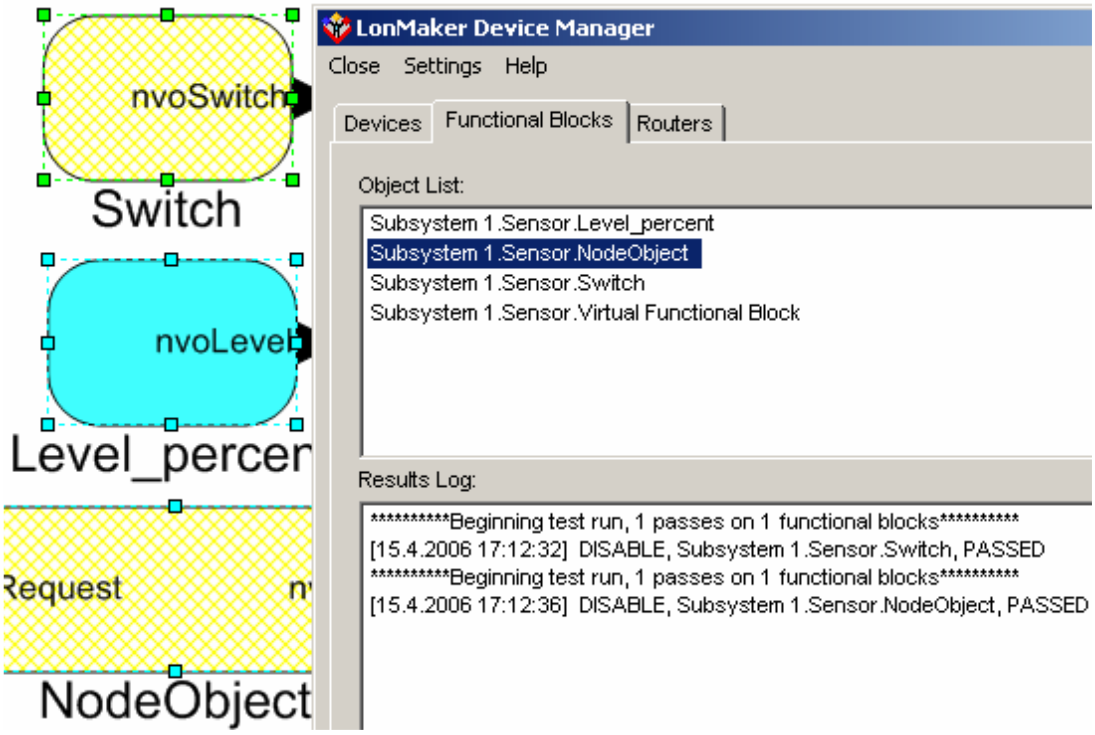
Kuva 45. Verkon fyysisen ja loogisen kerroksen kuvaukset.

Kun laitteeseen ”Sensor” on luotu kaikki toiminnalliset lohkot, on graafinen esitys kuvan 45. mukainen. Vision sisältämästä kuvan 46. esittämästä ”LonMaker”-työkalupalkista voidaan kytkeä pois ja päälle Vision esittämä verkon fyysinen, looginen ja datakerros. Näitä työkalupalkin nappeja painamalla voi tarkistaa, mitä kaikkia asetuksia verkkoon on jo tehty.

Kuva 46. Verkon fyysisen, loogisen ja datakerroksen graafisen esityksen kytkentä.

L5.2.5 Neuron-piirin sovelluksen testaus LonMaker Browserilla

Neuron-piiriin siirrettyä sovellusohjelmaa voidaan testata LonMakerin Browser-toiminnolla. Sillä voidaan lukea laitteen ulostuloverkkomuuttujien arvot sekä kirjoittaa sisääntuloverkkomuuttujiin. Sovelluksen verkkomuuttujien oikea toiminta voidaan näin testata, vaikka verkossa ei olisi muita laitteita testattavan laitteen ja verkkosovittimen lisäksi. Myös toiminnallisten lohkojen hallittavuus testataan.

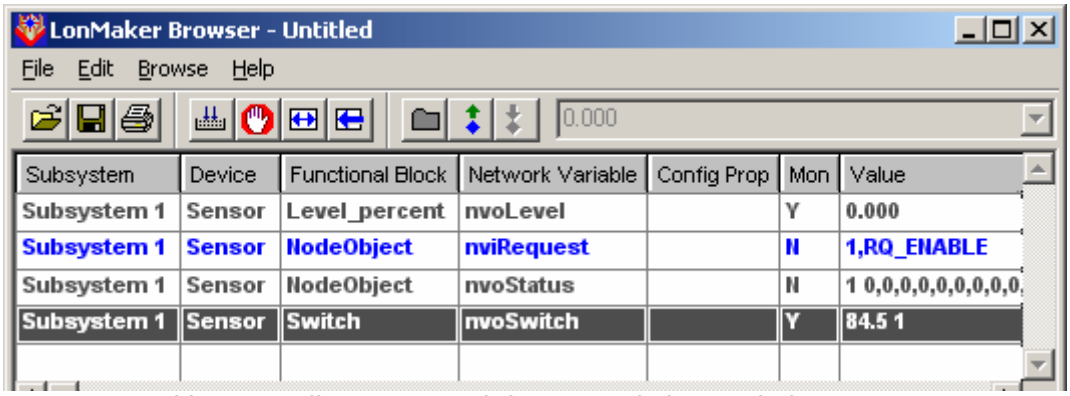


Kuva 47. Toiminnallisten lohkojen onnistunut kytkentä ilmaistaan tekstillä ”PASSED”.

Valitse kaikki toiminnalliset lohkot pitämällä shift-nappia pohjassa ja napsauttamalla hiirellä jokaista lohkoa. Valitse hiiren oikean näppäimen valikosta ”Manage”, jolloin kuvan 47. mukainen ikkuna aukeaa. Valitse hiirellä ”Object List”-ikkunasta yksi lohko kerrallaan ja paina Enable- sekä Disable-nappeja. Katso että jokaisen painalluksen jälkeen ”Results Log”-ikkunaan tulee teksti ”PASSED”. Ainoastaan virtuaalinen lohko ei tue kytkentää, muissa lohkoissa sen tulisi toimia. Visio-piirustuksessa passivoitu lohko esitetään keltaisella värillä.

Verkkomuuttujien arvojen testaamiseksi valitse Visio-piirustuksesta testattava laite tai lohko ja valitse hiiren oikean näppäimen valikosta ”Browse”. Kuvan 48. käyttötilanteessa tarkkaillaan verkkomuuttujien ”nvoLevel” ja ”nvoSwitch” arvoja. Muuttujan ”nvoSwitch” value-kentän arvoksi voidaan lukea 84.5 ja state-kentän arvoksi yksi. Lukeminen kytketään päälle joko napista ”Monitor All On” tai valitsemalla halutun verkkomuuttujan kohdalla avattavasta hiiren oikean näppäimen valikosta ”Monitor”. LonMaker kysyy tarkkailtavien ulostuloverkkomuuttujien arvoja säännöllisesti ja päivittää lukeman ikkunaan.

Vastaavasti sisääntuloverkkomuuttujille voidaan syöttää arvoja valitsemalla muuttuja hiirellä, kirjoittamalla luku Browser-ikkunan yläreunan kenttään ja painamalla enter-nappia.



Kuva 48. Verkkomuuttujien ”nvoLevel” ja ”nvoSwitch” arvoja luetaan.

Kun olet luonut ja testannut onnistuneesti verkkoon kirjoittavan ”Sensor”-sovelluksen, kytke pois päältä verkonhallinnan toimenpiteiden eteneminen fyysiseen verkkoon. Asetusten eteneminen on syytä katkaista, kun siirrytään luomaan uutta sovellusohjelmaa NodeBuilderiin. Näin estetään keskeneräisen sovelluksen tahaton latautuminen Neuron-piiriin. Asetus muutetaan valikosta LonMaker - Network Properties - Onnet/Offnet.

L5.3 Verkkoa lukevan solmun luominen

Työkurssin viimeisessä harjoitustyössä luodaan verkkoon kirjoittavan sovelluksen lisäksi verkkoa lukeva sovellus, ja luotujen Lon-verkon solmujen välinen looginen kytkeminen suoritetaan LonMaker-verkonhallintaohjelman graafisella käyttöliittymällä.

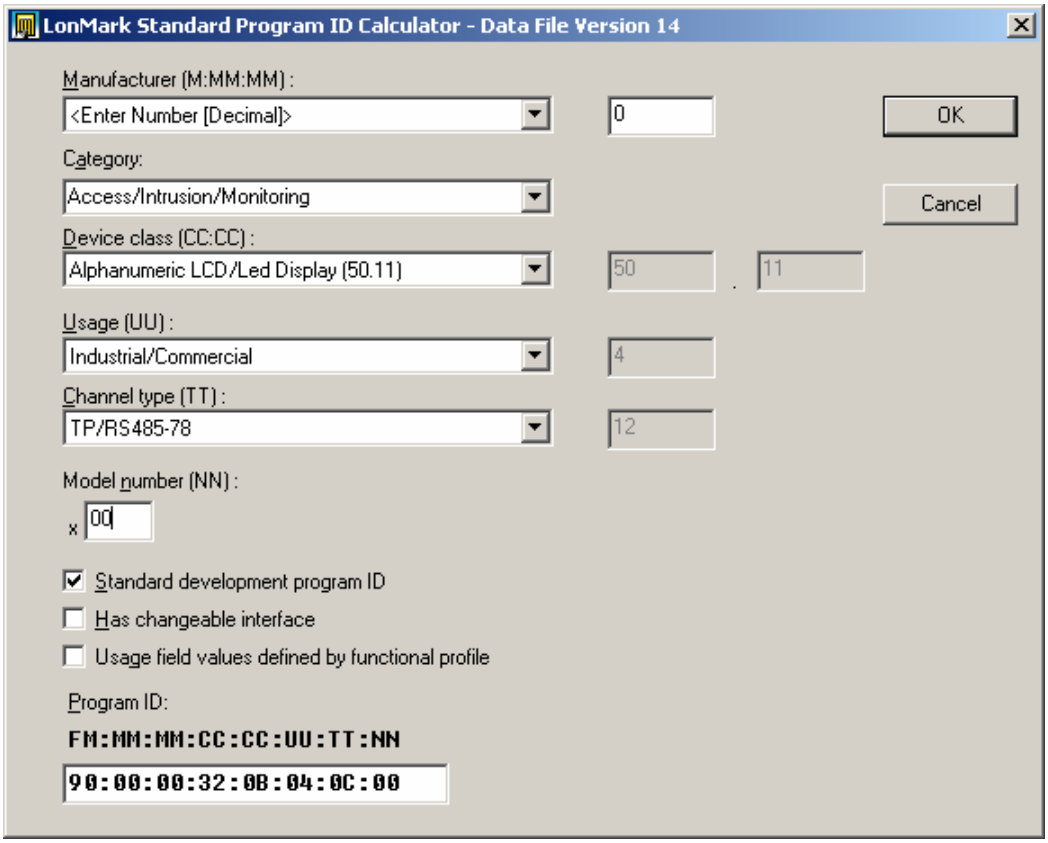
Verkkoa lukeva sovellus vastaanottaa sisääntuloverkkomuuttujaansa prosentteina ilmaistua tasosäätöä ja esittää vastaanottamansa arvon promillelukemana SPI-väylään kytketyllä LCD-näytöllä. Verkkoa lukeva sovellus siirretään simulointiympäristön kuvan 1. mukaisen erillisen osan oikean puoleiseen Neuron-piiriin.

Myös verkkoa lukevaan sovellukseen generoidaan kaksi itse määriteltyä toiminnallista lohkoa, joiden tyypiksi valitaan nyt ”SFTPopenLoopActuator”. Tämä tyyppi sisältää yhden pakollisen sisääntuloverkkomuuttujan. Toiselle lohkolle verkkomuuttujan tyypiksi määritellään jälleen ”SNVT_switch” ja nimeksi ”nviSwitch”. Toisen lohkon muuttujan tyypiksi valitaan ”SNVT_lev_percent” ja nimeksi ”nviLevel”.

Luo sovellusohjelmaa varten uusi projekti Nodebuilderiin. Aseta projektin verkoksi aiemmin luomasi LonMaker-verkko, joka sisältää jo ”Sensor”-solmun. Anna projektin nimeksi ”Actuator” ja luo sitä varten kotihakemistoosi polut ”\LonMaker\Actuator _source” ja ”\Lonmaker\ Actuator _output”. Anna laitemallin nimeksi ”Actuator”.

Laitemallin asetuksia tehdessäsi aseta ohjelman tunnuksen laskuriin kuvan 49. mukaiset arvot. Solmussa ”Actuator” on oltava erilainen ohjelman tunnus kuin solmussa ”Sensor”, jotta LonMaker myöhemmin erottaa solmuja varten luodut tiedostot toisistaan.

Mikäli NodeBuilder-projektia avattaessa on valittu LonMaker-verkko kuvan 7. mukaisesti, siirtää NodeBuilder sovellusta käännettäessä LNS-laitemallin LonMakerin verkon tietokantaan. Tämä tietokanta hyväksyy samalle ohjelman nimelle vain yhden tunnusnumeron, joten tunnusnumeron vaihtamiseksi on vanha LonMaker-verkko poistettava kuvan 2. esittämän käynnistysikkunan ”Delete”-napilla ja luotava uusi LonMaker-verkko.



Kuva 49. Sovellusohjelman tunnuksen laskuriin tehtävät asetukset.

L5.3.1 Verkkoa lukevan solmun ohjelmarungon täydennys

Actuator.h on sovellusohjelman päätiedoston otsikkotiedosto, joka sisältää globaaleja tyyppi- ja makromäärittelyjä. Sinne lisätään niiden ajastimien ja I/O-objektien määrittelyt, joiden tulee olla aina käytössä ja näkyä kaikkialle sovellusohjelmassa. Avaa tiedosto “pohja_actuator.txt” ohjelmointi-PC:n hakemistosta ”D:\Mallit” ja kopioi sieltä tarvittavat I/O- ja funktion prototyypin määrittelyt. Lisää otsikkotiedostoon määrittelyt omille globaaleille muuttujillesi, joiden avulla siirrät tiedot näytön päivittämiseksi lohkoilta pääohjelmaan.

Actuator.nc on sovellusohjelman päätiedosto. Se sisältää globaalit when-lauseet, joista annetaan reagointikomentoja halutuille toiminnallisille lohkoille niiden suuntaajafunktioita kutsumalla. Reagointi järjestelmätason tapahtumiin, esim. resetointiin, tapahtuu näiden when-lauseiden kautta. Kopioi mallitiedoston sisältämä näyttöä ohjaava funktio päätiedostoon. Muodosta oma when-lause, joka reagoi lohkojen vastaanottamaan tasosäätöön ja esittää tasosäätöarvon LCD-näytöllä. Koska tasosäätöarvot skaalataan verkkomuuttujille eri tavoilla, kannattaa skaalaus suorittaa verkkomuuttujan sisältävässä lohkossa.

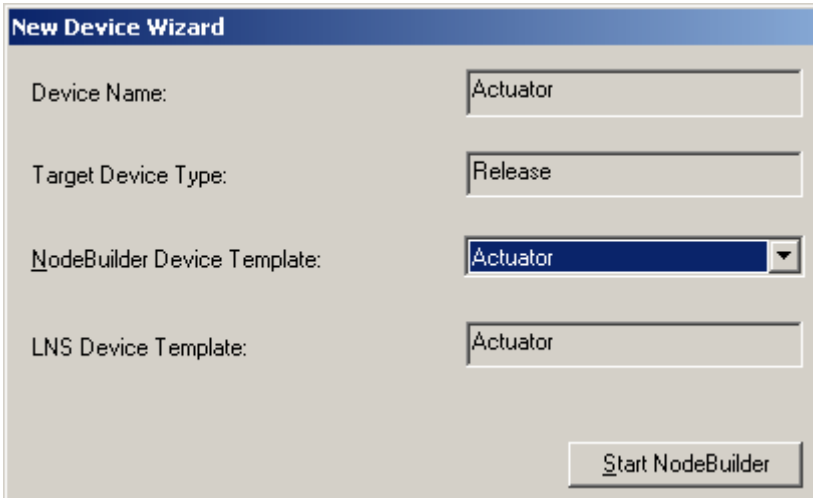
Luo lohkojen “Switch” ja “Level_percent” sisääntuloverkkomuuttujien päivittymiseen reagoiviin prosessilauseisiin ohjelmakoodit, jotka skaalaavat verkkomuuttujien arvot ja asettavat ne päätiedoston when-lauseen luettavaksi.

Tavoitteena on jälleen, ettei valmis sovellus reagoi molemman lohkon verkkomuuttujien syötteisiin yhtä aikaa. Lohkon ”Level_percent” vastaanottamaa verkkomuuttujan arvoa ei saa päivittää LCD-näytölle, jos myös lohkon ”Switch” havaitaan olevan aktiivisena. Tämä voidaan tarkastaa funktiokutsulla ”fblockNormalNotLockedOut(lohkon_nimi::global_index)”, joka palauttaa arvon TRUE, mikäli lohko on aktiivisena.

L5.3.2 Toisen solmun lisääminen LonMaker-verkkoon

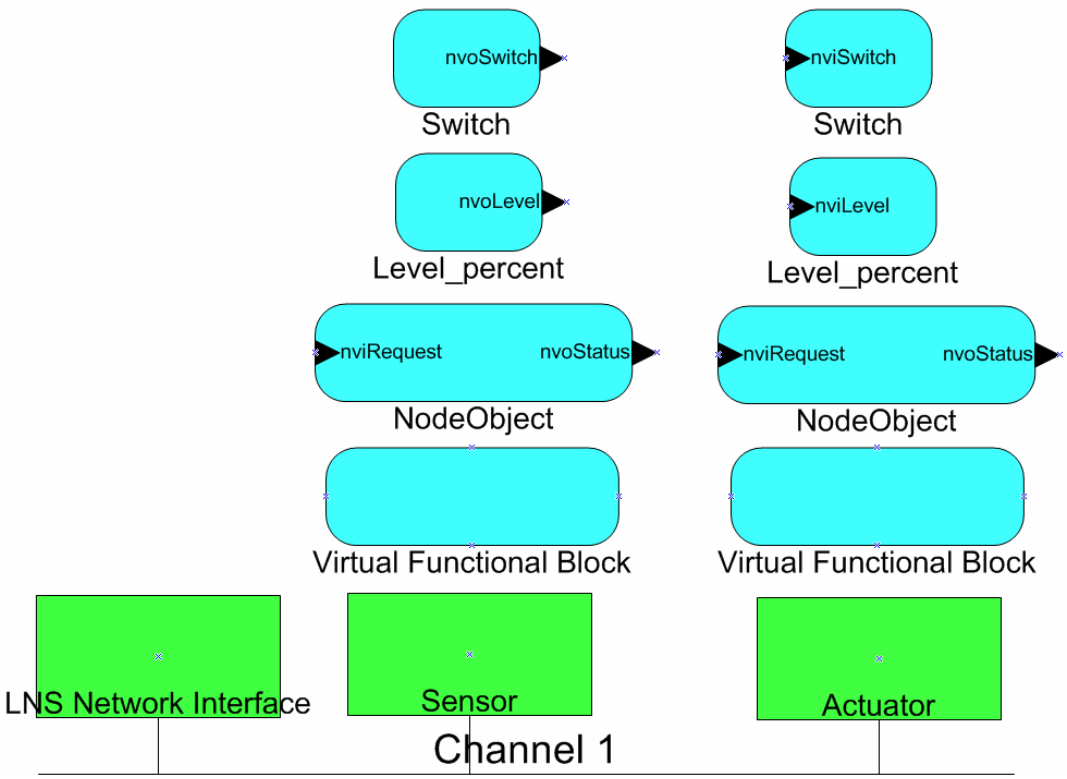
Onnistuneesti käännetty valmis sovellusohjelma siirretään Neuron-piiriin LonMaker-verkonhallintaohjelmalla. Avaa LonMakeriin aiemmin luomasi LonMaker-verkko, joka sisältää jo ”Sensor”-solmun. Aseta verkko Online-tilaan ja vedä hiirellä uusi laitemallin julkaisuversio Visio-piirrokseen. Anna uudelle laitteelle nimeksi ”Actuator” ja suorita sille toimeksianto (commission). Valitse laitemalliksi ”Actuator” kuvan 50. mukaisesti. Jollei sitä ole valittavana, paina kuvan oikean alakulman ”Start NodeBuilder”-nappia ja käännä sovellus ”Actuator” vielä kertaalleen NodeBuilderissa. Jätä tämä projekti auki NodeBuilderiin ja palaa LonMakeriin. Kuvan 50. valikosta pitäisi nyt löytyä laitemalli ”Actuator”.

Tarkista, että lähetin-vastaanottimen tyypiksi tulee TP/RS485-78, aseta piirin alkutilaksi ”Online” ja lataa juuri kääntämäsi binääri laitteiston oikeanpuoleiseen Neuron-piiriin.



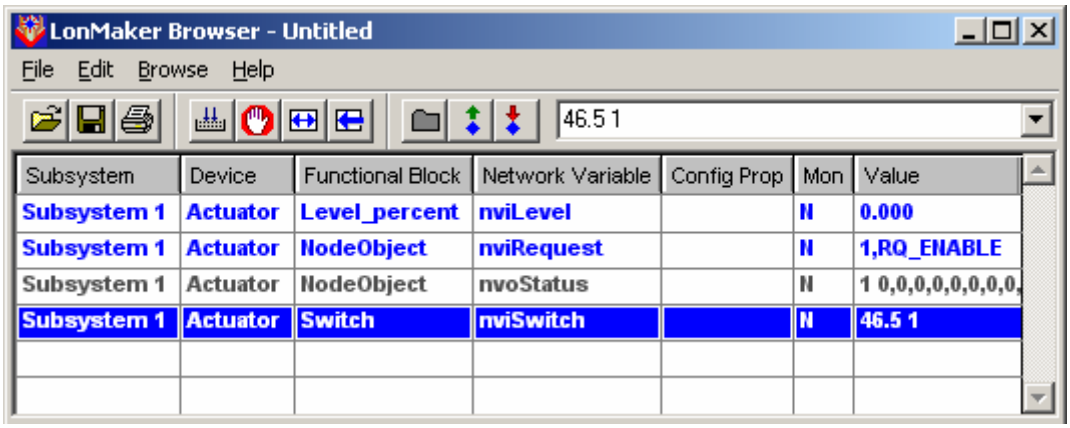
Kuva 50. Laitemallin valinta sovellusohjelman siirtoa varten.

Täydentääksesi verkon loogisen tason kuvauksen vedä taas hiirellä toiminnalliset lohkot Shapes-ikkunasta Visio-piirustukseen. Toiminnallisten lohkojen onnistuneen luomisen jälkeen Visio-piirustus on kuvan 51. mukainen.



Kuva 51. Verkon fyysisen ja loogisen kerroksen kuvaukset.

Solmun ”Actuator” toiminta testataan ensin erikseen LonMakerin Browser-toiminnolla syöttämällä sen sisääntulomuuttujiin arvoja ja havainnoimalla syötteiden aiheuttamia reaktioita LCD-näytöllä. Valitse Visio-piirustuksesta laite tai sen lohko ja valitse hiiren oikean näppäimen valikosta ”Browse”. Kuvan 52. käyttötilanteessa verkkomuuttujan ”nviSwitch” kenttään ”value” on syötetty arvo 46.5 ja kenttään ”state” arvo 1. Tällöin LCD-näytössä tulisi olla lukema ”465”. Luvut syötetään Browser-ikkunan yläreunan kenttään ja hyväksytään enter-napilla. Value- ja state-kenttien arvot erotetaan syötettäessä välilyönnillä.



Kuva 52. Verkkomuuttujaan ”nviSwitch” on syötetty arvo 46.5,1 (value, state).

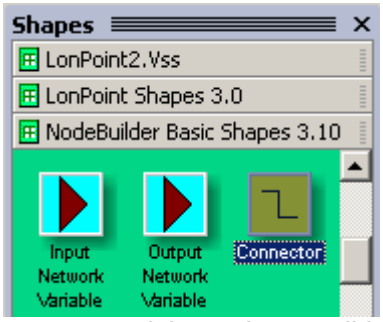
Kun luodut sovellukset on onnistuneesti testattu erikseen LonMaker Browser-toiminnolla, voidaan ne yhdistää loogisesti toisiinsa LonMaker-verkonhallintaohjelman graafisessa ympäristössä.

L5.4 Lon-verkon solmujen välinen looginen kytkeminen

Tässä harjoitustyössä on luotu ”Sensor”-solmu, joka kirjoittaa verkkoon prosentteina ilmaistavaa tasosäätöarvoa. Ulostuloverkkomuuttujan tyyppinä on ”SNVT_switch” tai ”SNVT_lev_percent” kytkettynä olevasta toiminnallisesta lohkosta riippuen. Lisäksi on luotu ”Actuator”-solmu, joka lukee tasosäätöarvoa jommankumman sisääntuloverkkomuuttujansa kautta. Nyt nämä solmut kytketään toisiinsa loogisesti LonMaker-verkonhallintaohjelman graafisella käyttöliittymällä.

Kytkennän aikana LonMaker kirjoittaa asetustietoja Neuron-piirin EEPROM-muistissa sijaitseviin verkkoliikenteen asetustaulukoihin. Se merkitsee osoitetaulukkoon kohdeosoitteeksi viestien kohteena olevan solmun loogisen Lon-verkon osoitteen. LonMaker asettaa yhdistettäville sisään- ja ulostuloverkkomuuttujille samat valitsintunnukset verkkomuuttujien asetustaulukkoon. Kaikilla loogisesti yhteen kytketyillä verkkomuuttujilla on sama valitsin, ja sen perusteella vastaanottaja osaa sijoittaa viestipaketin oikeaan sisääntuloverkkomuuttujaansa. Lisäksi LonMaker asettaa verkkomuuttujien välisessä viestiliikenteessä käytettävän LonTalk-protokollan palvelumuodon, joita voivat olla kuittaava, toistava tai kertaviesteihin pohjautuva yhteyspalvelu. Kaikki tämä tapahtuu automaattisesti eikä kytkentää suorittavan asentajan tarvitse puuttua näihin yksityiskohtiin, mikäli LonMakerin tekemät valinnat osoittautuvat tyydyttäviksi.

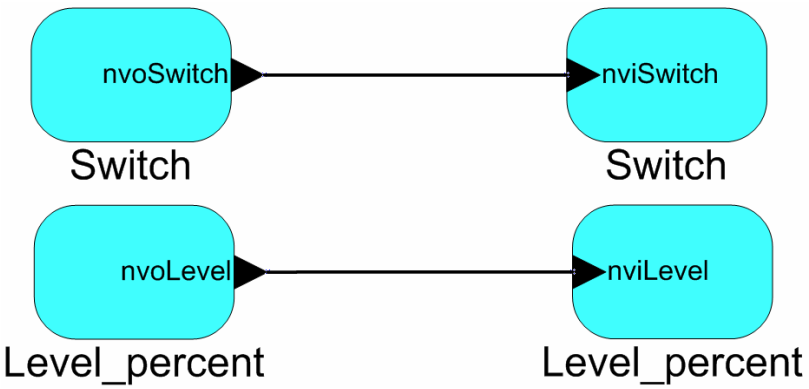
Kytkeäksesi verkkomuuttujat loogisesti vedä hiirellä kuvan 53. esittämä liitosjohdon kuvake kuvan 51. esittämään verkon graafiseen kuvaukseen. Tällöin Visio-piirustuksen ilmestyy kuvan 54. mukainen liitosjohto, joka sisältää erilaiset päät liitosten tekemiseen. Yhden sisääntulon ja yhden ulostulon välisessä kytkennässä ei liitosjohdon päiden järjestyksellä ole väliä. Useampien verkkomuuttujien välisessä liitoksessa yksi niistä valitaan keskukseksi, josta kytkentä tehdään kaikkiin muihin verkkomuuttujiin. Tällöin liitosjohdon keskittimenä toimiva pää kytketään aina keskusverkkomuuttujaan. Jos useamman verkkomuuttujan välisessä kytkennässä on vain yksi sisään- tai ulostulomuuttuja, tulee juuri se valita kytkennän keskukseksi.



Kuva 53. Vision ”Shapes”-ikkuna sisältää liitosjohdon kuvakkeen.

Kuva 54. Liitosjohdon toinen pää toimii keskittimenä (x) ja toinen kohteena (+).

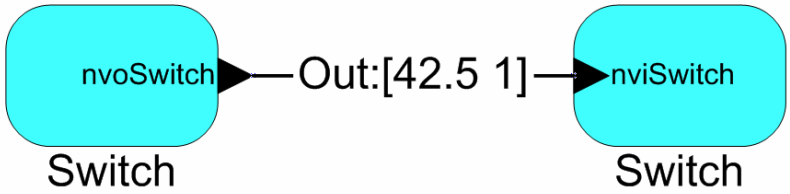
Kytke verkkomuuttujat ”nvoSwitch” ja ”nviSwitch” toisiinsa liitosjohdolla. Kun johdon molemmat päät on liitetty oikein, johdon väri muuttuu punaisesta mustaksi. Vedä ”Shapes”-ikkunasta toinen liitosjohto ja kytke sillä verkkomuuttujat ”nvoLevel” ja ”nviLevel”. Keskenään kytketyt toiminnalliset lohkot on esitetty kuvassa 55.



Kuva 55. Loogisesti kytketyt toiminnalliset lohkot ja verkkomuuttujat.

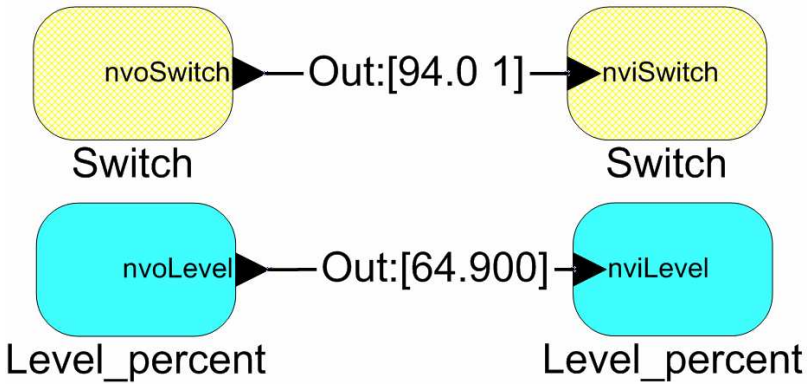
LonMakerin tekemiä verkkoasetuksia voidaan nyt tarkastella napsauttamalla laitteen fyysisen kuvauksen vihreää laatikkoa hiiren oikealla näppäimellä ja valitsemalla “Properties”. Välilehdestä ”Address Table” voidaan lukea Neuron-piirin osoitetaulukon sisältö ja välilehdestä ”Network Variable Config” on luettavissa verkkomuuttujakohtaisia asetuksia.

Napsauttamalla kytkettyä liitosjohtoa hiiren oikealla näppäimellä voidaan avautuvasta valikosta valita ”Monitor Output Value”, jolloin ulostuloverkkomuuttujan arvoa voidaan havainnollisesti seurata Visio-piirustuksesta kuvan 56. osoittamalla tavalla. Kuvassa verkkomuuttujan ”nvoSwitch” value-kentän arvoksi voidaan lukea 42.5 ja state-kentän arvoksi yksi.



Kuva 56. Verkkomuuttujan arvon seuranta Visio-piirustuksessa.

Tarkista vielä, että kaikkien lohkojen ollessa normaalissa toimintatilassa ainoastaan ulostuloverkkomuuttuja ”nvoSwitch” päivittyy. Vastaavasti kun ”Switch”-lohkot ovat pois kytkettyinä kuvan 57. mukaisesti, tulee ainoastaan ulostuloverkkomuuttujan ”nvoLevel” päivittyä. Kuvassa 57. verkkomuuttujalla ”nvoSwitch” on nollasta poikkeava arvo, mutta se ei muutu painonapeilla tasosäätöarvoa muutettaessa. Muutos tapahtuu ainoastaan verkkomuuttujassa ”nvoLevel”.



Kuva 57. ”Switch”-lohkot ovat pois kytkettyinä.

SUORITUS: Ryhmä varaa työskentelyajat simulointilaitteiston luona olevasta varauslistasta. Työaikoja saa varata korkeintaan kolmeksi tunniksi kerrallaan. Työtä suositellaan tehtäväksi määräaikoihin nähden etupainotteisesti, jotta vältytään ruuhkilta laitteiston käytössä.

Tehty harjoitus hyväksytetään assistentilla erikseen määrättyinä esittelyaikoina. Esittelyajat ja harjoituskohtaiset aikataululliset takarajat käyvät ilmi varauslistasta. Onnistuneen esittelyn jälkeen assistentille lähetetään sähköpostilla solmun ”Sensor” tiedostot ”sensor.nc” ja ”switch.nc” sekä solmun ”Actuator” tiedostot ”actuator.nc” ja ”level_percent.nc”. Sähköposti nimetään muotoon ”Ryhmä A, harjoitustyö B” ja varustetaan jäsenten nimillä ja opiskelijanumeroilla.

ARVOSTELU: Harjoituksen läpäisyyn vaaditaan vaihtoehtoisten rajapintojen toteuttaminen molemmissa solmuissa ja solmujen välisen viestinnän onnistuminen standardeja verkkomuuttujia käyttäen. Harjoituksesta voi saada kaksi suorituspistettä. Suorituspisteitä saa seuraavista ominaisuuksista:

- Valmiit sovellusohjelmat suorittavat tasosäätöarvojen skaalauksen virheettömästi. Solmun ”Sensor” lähettämät tasosäätöarvot rajoitetaan nollan ja 100 prosentin välille.
- Vaihtoehtoisten rajapintojen samanaikainen käyttö estetään onnistuneesti.

Kurssiarvosana määräytyy kerättyjen pisteiden perusteella taulukon 1. mukaisesti. Suorittamalla kaikista kurssin harjoitustöistä pelkän pakollisen osuuden saa arvosanaksi ykkösen. Viitosen saadakseen tulee kerätä vähintään 80 % suorituspisteistä. Pisteet ilmoitetaan prosentteina, sillä jos jonkin kurssin harjoitustyön suoritus estyy teknisistä syistä, lasketaan suoritettujen pisteiden osuus käytännössä tarjolla olleista maksimipisteistä.

Taulukko 1. Kurssiarvosanan määräytyminen.

Arvosana	1	2	3	4	5
Pisteet	<20 %	≥20 %	≥40 %	≥60 %	≥80 %

S-81.3210 Sulautettujen järjestelmien työkurssi (5 op)

Liite 6. SIMULOINTIYMPÄRISTÖN KÄYTTÖOHJE

Sulautettujen järjestelmien työkurssilla tutustutaan Neuron C -ohjelmointiin sekä Lon-verkkoihin hissin komponenttien välisessä viestinnässä. Työkurssi koostuu useammasta erillisestä harjoitustyöstä, jossa jokaisessa keskitytään simulointilaitteiston tiettyyn osaan. Kunkin harjoitustyön tavoitteena on tuottaa Neuron C -ohjelmakoodi tarkasteltavien komponenttien ohjaamiseen. KONE Oyj on lahjoittanut kuvan 1. mukaisen hissinohjausjärjestelmän simulointiympäristön kurssin käyttöön.

L6.1 Hissin toimintoja simuloiva laitteisto

Simulointilaitteisto koostuu nykyaikaisen hissiryhmän signalointiin käytettävästä ohjauselektroniikasta. Simulointiympäristön rakenne esitetään kuvassa 2. Laitteistossa ei ole mekaanisia liikkuvia osia, vaan hissikorien liikettä hissikuiluissa simuloidaan pöytäsimulaattoreilla. Pöytäsimulaattori sisältää hissi-CPU:n, joka tyypillisesti sijaitsee hissikuilussa ja ohjaa siinä kulkevan hissikorin liikettä. Hissikorin paikkatietoa ylläpidetään aina hissi-CPU:ssa, ja pöytäsimulaattorissa paikkatieto luodaan simulointiin räätälöidyllä hissi-CPU:n ohjelmistolla.

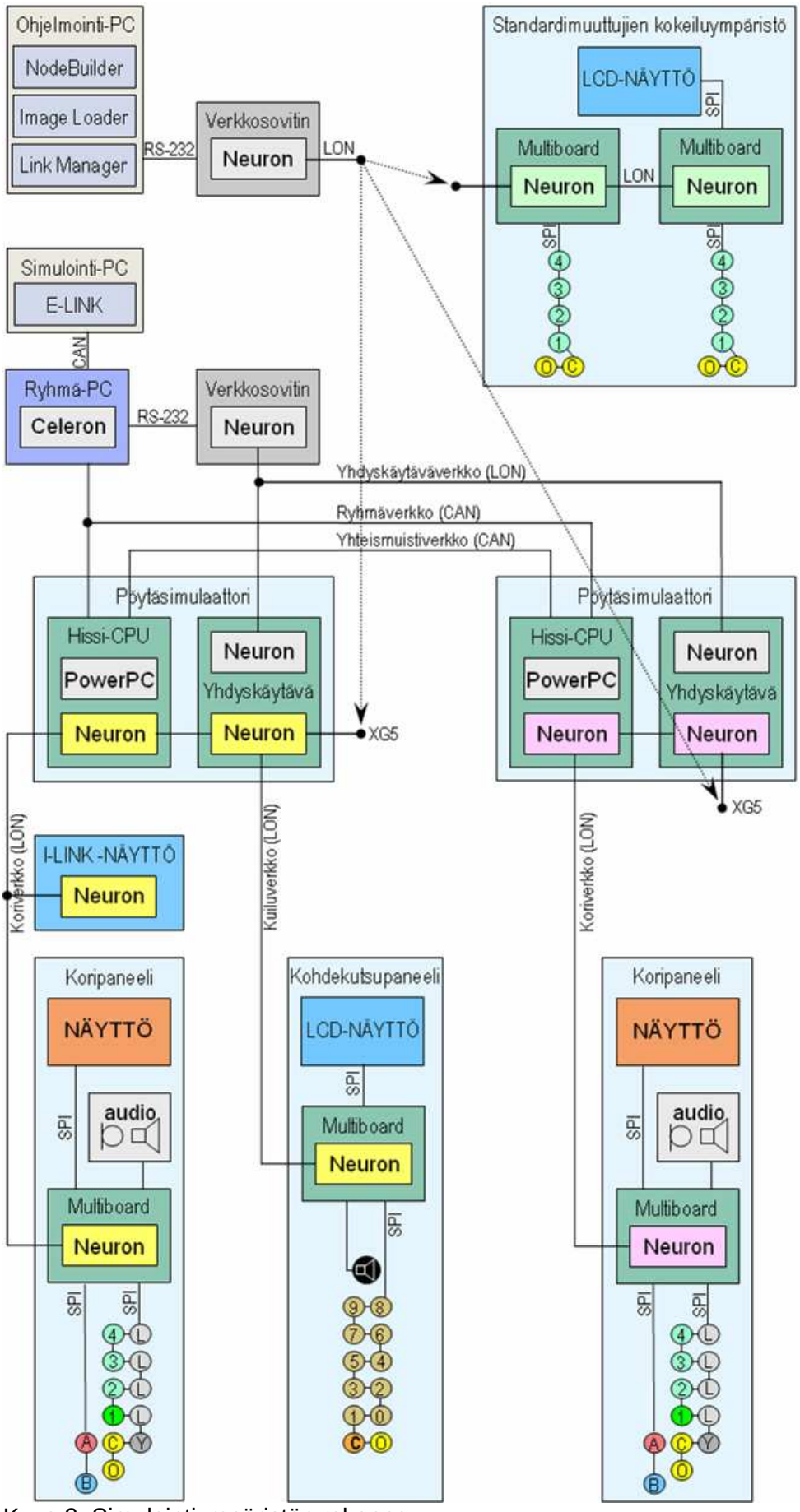
Laitteisto muodostaa kahden hissin ryhmän, joita ohjaa erillinen tietokone eli ryhmä-PC. Kahden hissi-CPU:n muodostama ryhmä kykenisi toimimaan ilman ryhmä-PC:n ohjausta, mikäli ulkokutsut annettaisiin perinteisellä tavalla ylös- tai alaspäin. Simulointiympäristö sisältää kuitenkin uudentyyppisen kohdekutsupaneelin, ja ryhmä-PC:tä tarvitaan tulkitsemaan sen antamia kutsuja.

Kohdekutsupaneeli sijoitetaan tyypillisesti rakennuksen aulaan, jonka kautta suurin osa hissin käyttäjistä kulkee. Aulasta käyttäjä tilaa hissin näppäilemällä kohdekutsupaneeliin sen kerroksen numeron, johon hän on matkalla. Tällöin ryhmä-PC valitsee hissiryhmästä suunnan ja etäisyyden perusteella sopivimman hissin palvelemaan tätä kutsua. Ryhmä-PC palauttaa kohdekutsupaneelille kuljetustiedot, joilla matkustaja opastetaan oikean hissin ovelle. Aulan lisäksi muissakin kerroksissa voi olla kohdekutsupaneelit, tai niissä voidaan käyttää perinteisiä ylös/alas-kutsunappeja. Kurssin simulointiympäristössä ulkokutsuja annetaan vain yhdellä kohdekutsupaneelilla.

Kurssilla kehitetään sovellusohjelma kohdekutsupaneelin lisäksi koripaneelin Neuron-piirille. Molempiin pöytäsimulaattoreihin liittyy täysin varusteltu hissikorin paneeli, josta annetaan korikutsuja ja lukitaan kerroksia avaimilla. Käyttäjä saa palautetta näytön ja nappien indikaattorivalojen kautta.

Varsinaisen hissin toimintaa simuloivan osan lisäksi laitteisto sisältää erillisen kahden Neuron-piirin ympäristön. Muusta laitteistosta riippumaton osa mahdollistaa kahden Lon-verkon solmun toiminnan vapaan määrittelyn. Siinä harjoitellaan sovellusviestien lähettämistä ja vastaanottamista sekä tutustutaan standardin verkkoliikenteen rajapinnan luomiseen.

Kuva 1. Kurssin opetuksessa käytettävä laitteisto.



Kuva 2. Simulointiympäristön rakenne.

L6.1.1 Koripaneelin Multiboard

Kaikki kurssilla luotavat Neuron C -sovellukset ladataan Multiboard-piirilevylle. Se on ensisijaisesti suunniteltu käytettäväksi hissin koripaneelissa, mutta nimensä mukaisesti sitä voidaan käyttää muissakin kohteissa Neuron-piirin sovellusohjelmaa räätälöimällä.

Multiboardin sisältämä Neuron-piiri on Toshiban valmistaman TMPN3120FE5M, joka sisältää 16 Kb:n ROM-muistin varusohjelmistolle sekä 4 Kb SRAM- ja 3 Kb EEPROM-muistia. RAM-muistia on enemmän kuin muissa Neuron-piireissä, joten resurssit loppuvat tyypillisesti ensimmäisenä sovellusohjelman ja asetustietojen säilytykseen käytettävästä EEPROM-muistista.

Simulointiympäristön käyttämä Lon-verkon siirtomedia on kierretty pari ja siirto tapahtuu RS485-sarjaliikennestandardin mukaisesti siirtonopeudella 78 kb/s. Multiboard sisältää tähän siirtokanavaan sopivan lähetin-vastaanottimen, jota Neuron-piiri ohjaa mm. sovellusohjelman sisältämillä asetusparametreilla.

Multiboardissa tärkeimpiä paikallisia I/O-toimintoja ohjataan jaetun SPI-väylän kautta. Jokaiselle SPI-väylää käyttävälle toiminnolle on oma Neuron-piirin I/O-pinni, jolla väylä varataan kyseisen toiminnon käyttöön. Kutsunapeille ja näytölle on omat SPI-väylän liittimensä. Lisäksi Multiboard sisältää valmiuden hälytys- ja äänitoiminnoille.

Koripaneelissa Multiboardin ja sen sisältämän Neuron-piirin tärkein tehtävä on viestien kääntäminen Lon-koriverkon ja SPI-väylän välillä. Kutsunapeilla hissikorista lähetetään palvelupyyntö kuilussa sijaitsevalle hissi-CPU:lle, jonka vastaus esitetään kutsunappien indikaattorivaloilla ja koripaneelin näytöllä. Neuron-piirin ohjelmakoodiin voidaan lisäksi sisällyttää koripaneelin omaa toimintalogiikkaa ja hissikorikohtaisia asetuksia.

L6.1.2 Multiboard kohdekutsupaneelissa

Simulointiympäristön kohdekutsupaneeli sisältää samanlaisen Multiboard-piirilevyn kuin koripaneeli, ainoastaan Neuron-piirin sovellusohjelma on erilainen. Päätehtävänä on edelleen viestien kääntäminen Lon-verkon ja SPI-väylän välillä, mutta verkkomuuttujien mahdollistaman automaattisen viestinlähetyksen sijasta kohdekutsuviestit rakennetaan ja lähetetään sovellusohjelmassa eksplisiittisesti. Lähetettävien viestien kohteena on ryhmä-PC, jolle viestit kulkevat yhdyskäytävän kautta. Tämä huomioidaan viestin osoite- ja tunnustietojen määrittelyssä.

Kohdekutsupaneeli sisältää LCD-näytön, jonka ohjauskäskyt eroavat koripaneelin pistematriisinäytön ja merkkilamppujen yhdistelmästä. Lisäksi näppäinyhdistelmänä annettava kohdekutsu edellyttää toimintalogiikan rakentamista sovellusohjelmaan.

L6.1.3 Yhdyskäytävä

Asetus- ja paikkatietoja lukuun ottamatta hissiryhmän muodostavat hissit ovat tyypillisesti identtisiä. Jokaista hissikoria ohjaa oma hissi-CPU, ja hissien välisen yhteistoiminnan järjestämiseksi ne kommunikoivat keskenään. Tässä viestinnässä käytetään peruskokoonpanossa CAN-väylään pohjautuvaa ryhmäverkkoa, ja vaativammissa kokoonpanoissa ryhmäverkkoon kytketään myös ryhmä-PC. Ryhmä-PC ottaa vastuun hissien välisen yhteistoiminnan organisoinnista, ja sen viestinnän hoitamiseksi hissiryhmään lisätään yhdyskäytäväverkko. Hissi-CPU:t saavat tällöin kylkeensä yhdyskäytäväkortin, johon yhdyskäytävä- ja kuiluverkot kytketään.

Yhdyskäytävä jakaa Lon-verkon kanaviin ja suodattaa kanavien välistä liikennettä. Se reitittää ryhmä-PC:ltä lähtevät viestit oikeille hisseille ja estää hissien sisäisen

viestinnän etenemisen yhdyskäytäväverkkoon. Näin se jakaa Lon-verkkoa liikennekuormituksen kannalta järkeviin osiin. Yhdyskäytävä sisältää Neuron-piirin ja lähetin-vastaanottimen kummallekin muodostamalleen kanavalle. Näin se vahvistaa jokaisen välittämänsä signaalin ja lisää Lon-verkon ulottuvuutta ja luotettavuutta.

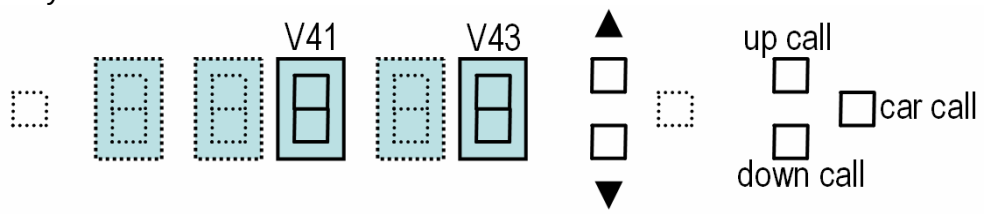
Verkonhallintaohjelmalla ei nähdä yhdyskäytävän toisella puolella olevia Neuron-piirejä. Yhdyskäytävän sisältämistä Neuron-piireistä nähdään se, kumman puoleista Lon-verkon osaa verkonhallintaohjelmalla tutkitaan. Simulointiympäristön rakennetta esittävään kuvaan 2. on merkitty samalla tunnusväreillä ne Neuron-piirit, jotka pystytään havaitsemaan samasta Lon-verkon liittymäkohdasta. Päälaitteiston liittymäkohdat on merkitty tunnuksella ”XG5”.

L6.1.4 Hissi-CPU

Jokaista hissikoria ohjaa oma hissi-CPU, joka oikeassa käyttöympäristössä ollessaan sijaitsee hissikuilussa. Hissi-CPU kykenee itsenäiseen toimintaan ja vastaa viime kädessä ohjaamansa hissikorin liikkeistä ja toimintalogiikasta. Se vastaanottaa hissikorista annetut korikutsut sekä kerroksista hissin ulkopuolelta annetut tasokutsut. Ryhmä-PC:n läsnä ollessa se välittää tasokutsut, muuten hissi-CPU vastaanottaa ne omasta Lon-kuiluverkostaan tai palvelupyyntönä CAN-väylää pitkin toiselta hissi-CPU:lta.

Kutsun vastaanotettuaan Hissi-CPU lähettää hissikorille viestit koripaneelin kutsunapin indikaattorivalon sekä näytön ohjaamiseksi. Hissi-CPU vastaa koripaneelin näytön sisällöstä ja kerroslukitusten toimintalogiikasta, koripaneelin Neuron-sovelluksen tehtävänä on tässä vain viestien ohjaaminen Lon-verkon ja SPI-väylän välillä.

Hissi-CPU:lta voidaan antaa kori- ja tasokutsuja kuvan 3. esittämiä nappeja käyttäen. Näyttö V41 ilmoittaa jatkuvasti hissikorin sijaintikerroksen. Nuolinäppäimillä voidaan näytölle V43 hakea jokin toinen kerrosnumero, ja tällöin voidaan antaa kutsu painamalla jotakin call-nappia. Car call antaa korikutsun määriteltyyn kerrokseen ja up/down call antaa tasokutsun määritellystä kerroksesta ylös tai alas siirtymiseen.



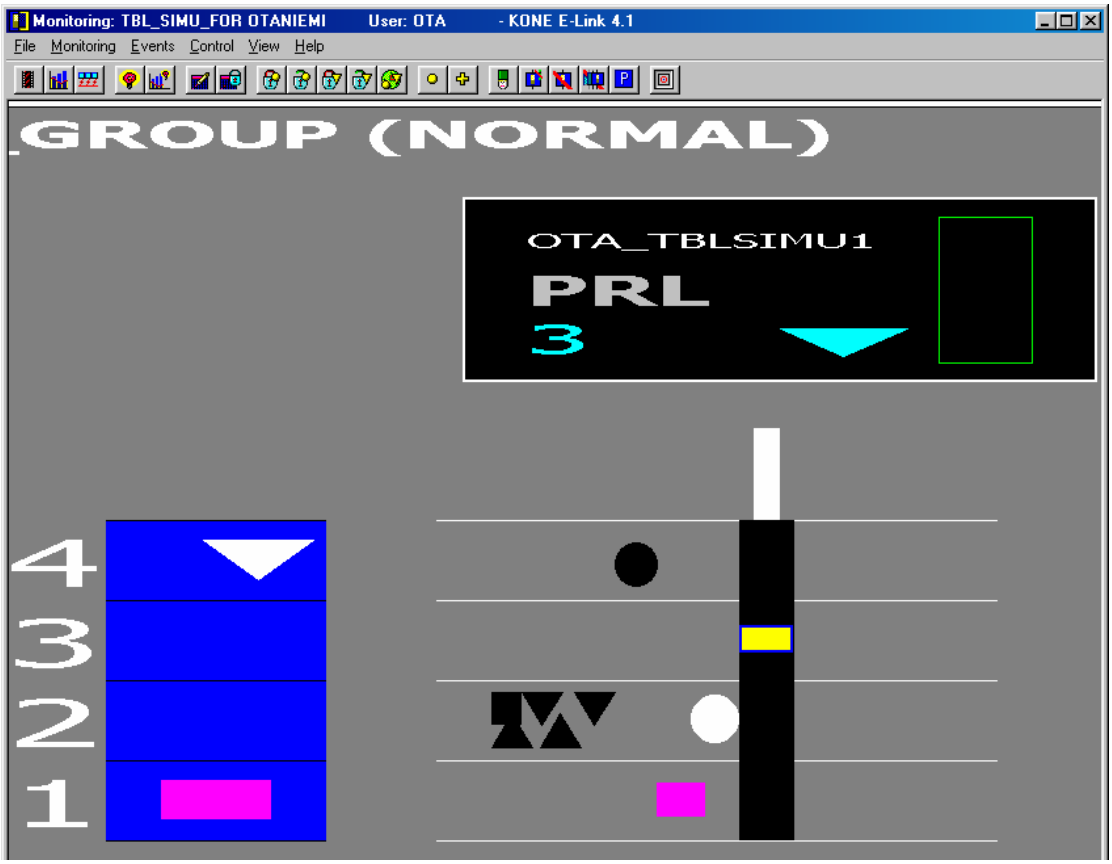
Kuva 3. Kori- ja tasokutsujen antoon tarvittavat napit ja näytöt hissi-CPU:ssa.

Kuvan 3. esittämä hissi-CPU:n näyttö antaa jo perustiedon hissikorin sijainnista, ja lisäinformaatiota simuloidun hissikorin liiketilasta ja ajosuunnitelmasta voi saada kutsunappien valoista ja koripaneelin näytöstä. Myös ensimmäisen hissin koriverkkoon kytketty I-Link -näyttö kertoo hissikorin sijainnin ja kohteen. I-Link -näyttö sijaitsee tyypillisesti hissikorissa ja esittelee hissikorin sijaintikerroksen liiketiloja. Selkeimmän kokonaiskuvan simulointiympäristön hissien toimintatiloista tarjoaa kuitenkin E-Link-valvontaohjelma.

L6.1.5 E-Link

Hissiryhmän yhteistoiminnan mahdollistavaan CAN-ryhmäverkkoon on kytketty simulointi-PC. Sen ruudulta voidaan havainnollisesti seurata hissikorien liikettä hissikuiluissa. Tämän mahdollistaa E-Link -ohjelma, jolla valvotaan ja hallitaan hissiryhmää.

E-Link esittää graafisesti hissikorien sijainnit hissikuiluissa, ovien toiminnan sekä annetut kori- ja tasokutsut. Myös erikoistoiminnot näkyvät E-Linkin ruudulla, esim. lukitut kerrokset ja prioriteettiajo.



Kuva 4. E-Link-ohjelman käyttötilanne.

Kuva 4. esittää E-Link-ohjelman käyttötilanteen. Yläreunassa on osittain peitossa hissiryhmän nimi, OTA_GROUP. Sen perässä on suluissa ryhmän toimintatila, NORMAL. Tässä hissiryhmässä on vain yksi hissi, OTA_TBLSIMU1.

Mustapohjaisesta laatikosta voidaan hissin toimintatilaksi lukea PRL, eli se on juuri siirtymässä palvelemaan prioriteettitasokutsua. Mustasta laatikosta voidaan myös lukea hissikorin sijainniksi 3. kerros ja suunnaksi alaspäin.

Vasemmalla oleviin sinisiin laatikoihin tulee merkintä hyväksytyistä tasokutsuista, tässä 4. kerroksesta on annettu kutsu alas siirtymiseen ja 1. kerroksesta on annettu prioriteettikutsu.

Hissikuilun graafisen kuvauksen viereen tulee kerroskohtaisia merkintöjä. Mikäli hissi ei ole läpikuljettavaa mallia eli siinä on vain yksi ovi, tulevat kaikki merkinnät kuilun vasemmalle puolelle. Valkoinen pallo ilmaisee korikutsua ja violetti neliö prioriteettitasokutsua. Musta pallo ilmaisee, ettei kerrokseen pääse normaalilla korikutsulla. Mustien kolmioiden ja neliön ryhmä tarkoittaa, ettei kyseisestä

kerroksesta voida tilata hissiä. Merkinnän sisältämä kaksinkertainen määrä kolmioita viestittää, että lukitus koskee koko kerrosta.

Jotta ohjelmalla voidaan antaa kutsuja ja tehdä asetuksia, tulee kirjautua sisään File-valikosta. Käyttäjätunnuksena ja salasanana on ”ota”. Kutsujen antamiseksi klikataan ensin ”Car call” tai ”Priority landing call” nappia, jolloin kursori vaihtuu kyseisen kutsun symboliksi. Sitten tällä erikoiskursorilla klikataan haluttua kerrosta. Vastaavalla tavalla voidaan lukita kerroksia ja asettaa halutun hissin toimintatila. Tarjolla olevat toimintatilat on selitetty seuraavassa lyhenneluettelossa.

E-Link ilmaisee hissiryhmän toimintatilan tunnustekstillä, joka on sulkeissa heti hissiryhmän nimen jälkeen. Hissiryhmän toimintatiloja:

NOC = No connection, valvontaohjelmalla ei ole yhteyttä hissiryhmään. Joko hissiryhmä on pois päältä tai viestintäyhteys ei toimi.

NORMAL = Hissikorit liikennöivät normaalisti ja kaikkia kutsuja palvellaan tasaveroisesti, ellei prioriteettia ole erikseen määritelty.

Yksittäisen hissin toimintatila ilmaistaan hissikuilun graafisen esityksen yläpuolella olevassa mustassa laatikossa seuraavilla lyhenteillä:

NOR = Normal drive.

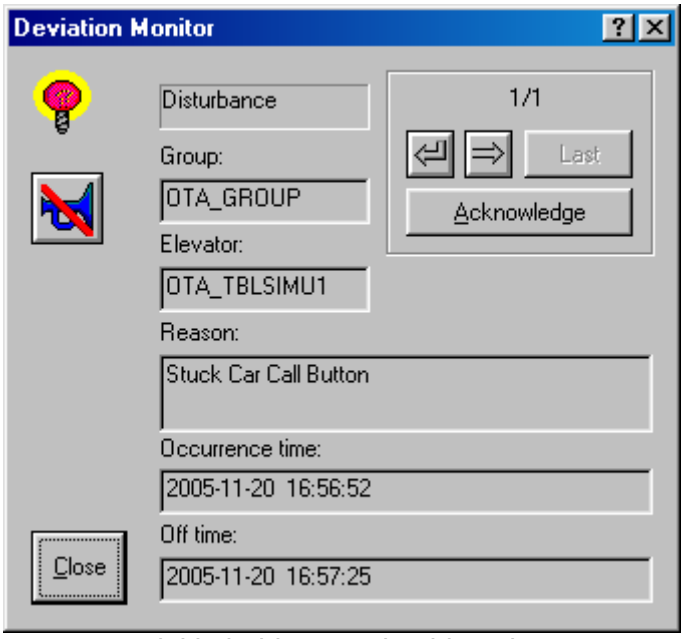
ATS = Attendant service, hissipoika ajaa hissiä manuaalisesti.

PRC = Independent service, hissi vastaanottaa kutsuja ainoastaan hissikorista ja jättää kerroksista annetut kutsut huomioimatta.

PRL = Priority call at landing, hissi antaa etusijan tasokutsulle ja jättää muut kutsut huomiotta.

SIM = Simplex service, hissi voidaan irrottaa muusta hissiryhmästä, jolloin se palvelee itsenäisesti annettuja tasokutsuja.

PAD = Special parking (parking at predefined door), hissin vapautuessa se ajetaan aina määriteltyyn kerrokseen. Kerrokseksi tulee se, jossa toiminto alun perin aktivoidaan.



Kuva 5. E-Linkin häiriöraportti pohjaan juuttuneesta napista.

E-Link antaa myös raportin hissin toimintahäiriöistä, esim. jos koripaneelin kutsunappi juuttuu pohjaan. Kuva 5. esittää häiriöraportin tilanteesta, jossa nappi on ollut jumiutuneena 33 sekunnin ajan.

Simulointi-PC sisältää ISA-väyläisen CAN-verkkosovittimen ja Windows98-käyttöjärjestelmän, minkä takia sillä ei ole muita tehtäviä simulointiympäristössä, vaan ohjelmointia varten käytämme erillistä nykyaikaista PC:tä.

L6.2 Verkonhallinta- ja ohjelmointityökalut

Verkonhallinta- ja ohjelmointityökalut toimivat erillisessä ohjelmointi-PC:ssä, joka kytketään verkkosovittimen välityksellä simulointiympäristön Lon-verkkoon. Ohjelmointityökaluna toimii NodeBuilder, jonka editorilla ohjelmakoodi kirjoitetaan. NodeBuilder myös kääntää kirjoitetun koodin binääriksi, joka voidaan siirtää Neuron-piiriin.

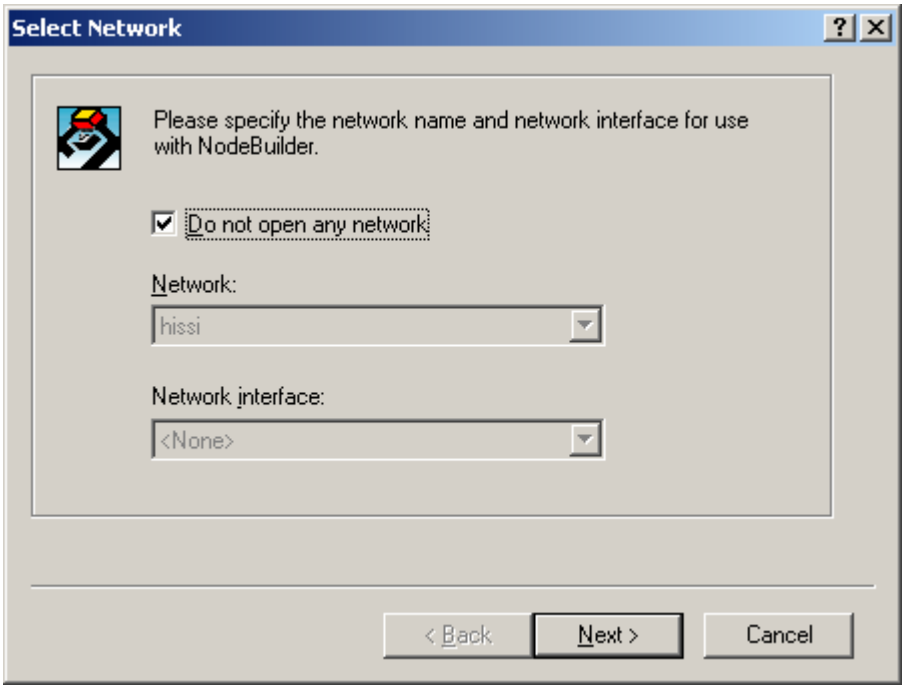
Kurssi tutustuttaa opiskelijat kahteen erilaiseen maailmaan: Valmistajakohtaiseen Lon-verkkoon sekä eri valmistajien tuotteiden keskinäistä yhteensopivuutta edistäviin standardeihin verkkomuuttujiin ja laiteprofiileihin. Tämä heijastuu kurssilla käytettäviin verkonhallintatyökaluihin. Verkonhallintatyökaluina on kaksi vaihtoehtoista ohjelmaa, KONE Oyj:n käyttämä Lon Node Image Loader ja Echelonin LonMaker, joka on osa NodeBuilder-ohjelmistopakettia. Ensin mainittua käytetään liityttäessä hissin toimintoja simuloivaan laitteistoon, jolloin pitäydytään laitetoimittajan käyttämissä työkaluissa. Kurssilla on lisäksi käytettävissä erillinen laitteisto standardien verkkomuuttujien käyttöön. Tällöin hyödynnetään NodeBuilder-ohjelmistopaketin ominaisuuksia laitteiden liityntäpintojen luomiseksi. Harjoitustyössä päästään kokeilemaan standardin liityntäpinnan toteuttavan ohjelmakoodin automaattista generointia ja luotujen Lon-verkon solmujen yhdistämistä LonMaker Integration Tool -ohjelman graafisessa ympäristössä.

Saadakseen yhteyden Lon-verkkoon verkonhallintatyökalu tarvitsee palveluja verkkosovittimen ajurilta. Tätä SLTALink Manager -ohjelmaa ajetaan taustalla ohjelmointi-PC:ssä. Ohjelman nimi tulee verkkosovittimesta, Serial LonTalk Adapter, joka kytkeytyy ohjelmointi-PC:een sarjaporttia pitkin.

L6.2.1 NodeBuilder

Varsinaisena ohjelmointityökaluna kurssilla toimii Echelonin NodeBuilder, jonka editorilla ohjelmakoodi kirjoitetaan. NodeBuilder myös kääntää kirjoitetun koodin binääriksi, joka siirretään Neuron-piiriin verkonhallintatyökaluilla.

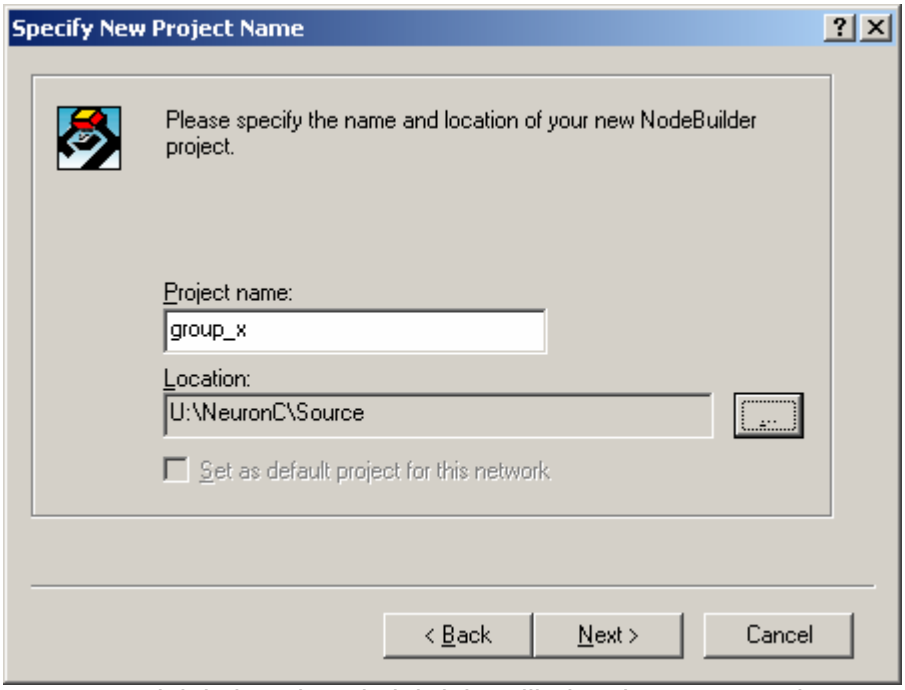
Käynnistettäessä NodeBuilderia ensimmäistä kertaa valitaan ”File”-valikosta ”Create Project”, myöhemmin voidaan valita ”Open Project”. Molemmissa tapauksissa NodeBuilder tiedustelee kuvan 6. mukaisesti käytettävää verkkoa. Käynnistettäessä NodeBuilder hissiverkon harjoitustöitä varten ei NodeBuilderin ole tarkoitus nähdä Lon-verkkoa lainkaan. Sillä vain luodaan ohjelmakoodi ja käännetään se, ja binäärin siirto ja muu verkonhallinta tehdään Lon Node Image Loaderilla. Tällöin käytetään kuvan 6. mukaista asetusta.



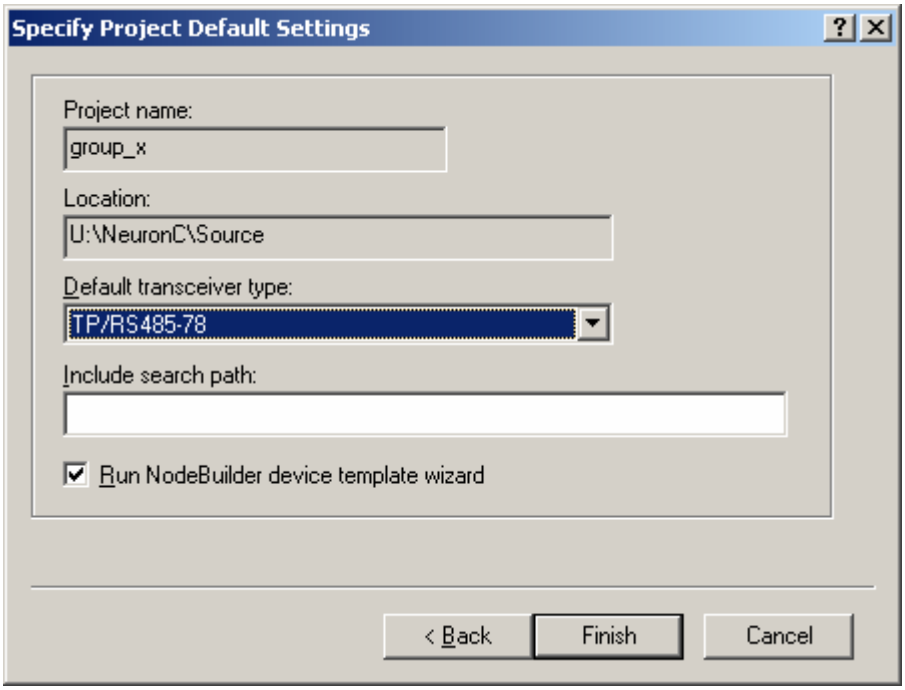
Kuva 6. NodeBuilderin käynnistäminen hissiverkkoprojektia varten.

Harjoitustöitä varten jokainen kurssin ryhmä luo NodeBuilderilla oman projektinsa ja tallettaa sen kotihakemistoonsa. Jokaiselle ryhmälle on käyttäjätunnus ja salasana simulointiympäristön ohjelmointi-PC:lle. Käyttäjätunnus on muotoa ”group_x”, jossa x on ryhmän numero. Oman ryhmäkohtaisen salasanan saa kurssin assistentilta.

Luo lähdekoodeja varten omaan kotihakemistoosi polku ”\NeuronC\Source” sekä käännettyjä binääritiedostoja varten hakemistopolku ”\NeuronC\Output”. Projektia luodessasi anna sen nimeksi ”group_x”, jossa x on ryhmänne numero. Aseta kotihakemistosi polku ”\NeuronC\Source” lähdekoodien hakemistoksi kuvan 7. mukaisesti.

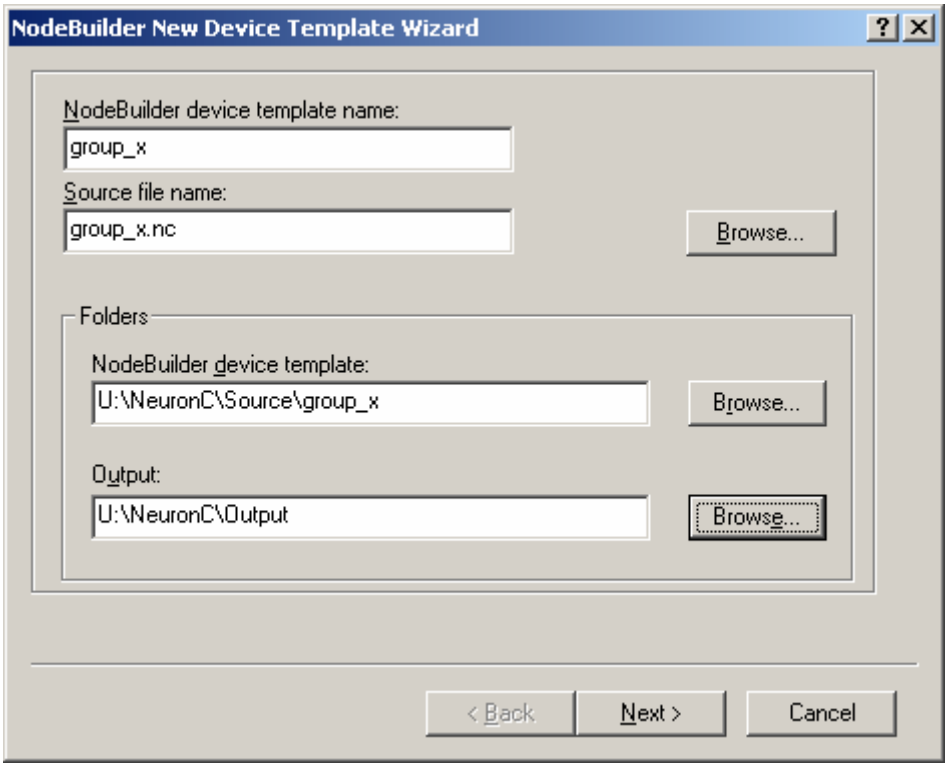


Kuva 7. Projektin luominen ja lähdekoodihakemiston asettaminen.

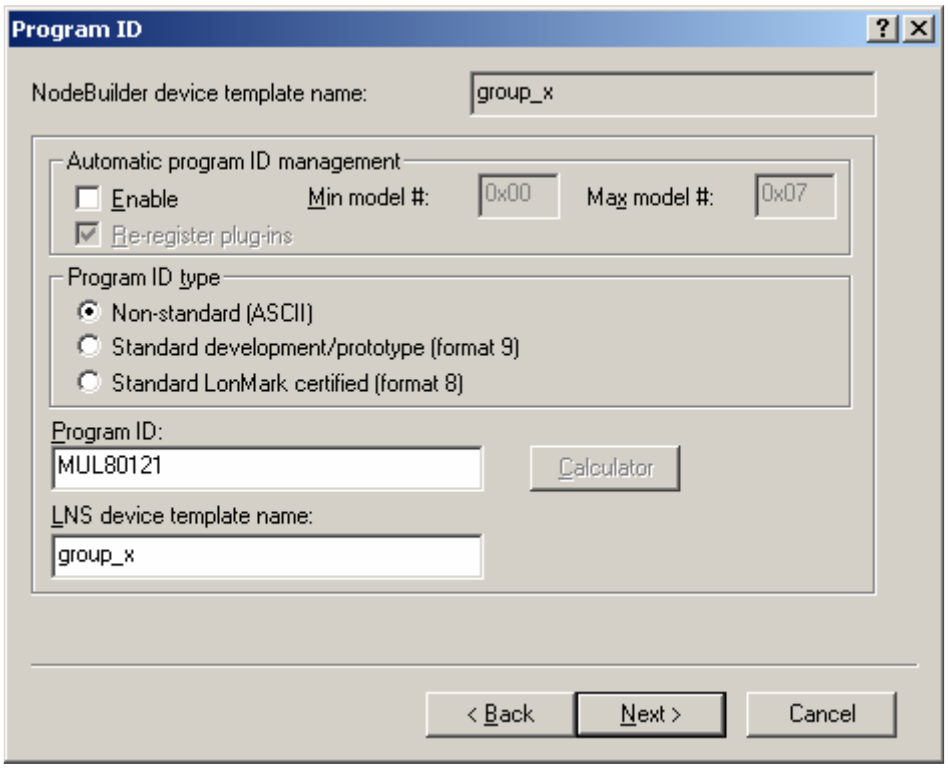


Kuva 8. Valitse lähetin-vastaanottimeksi TP/RS485-78.

Simulointiympäristön Lon-verkon siirtomediana toimii kierretty pari (Twisted Pair), siirto tapahtuu EIA:n (Electronics Industry Association) RS485-sarjaliikennestandardin mukaisesti ja siirtonopeutena on 78 kb/s.

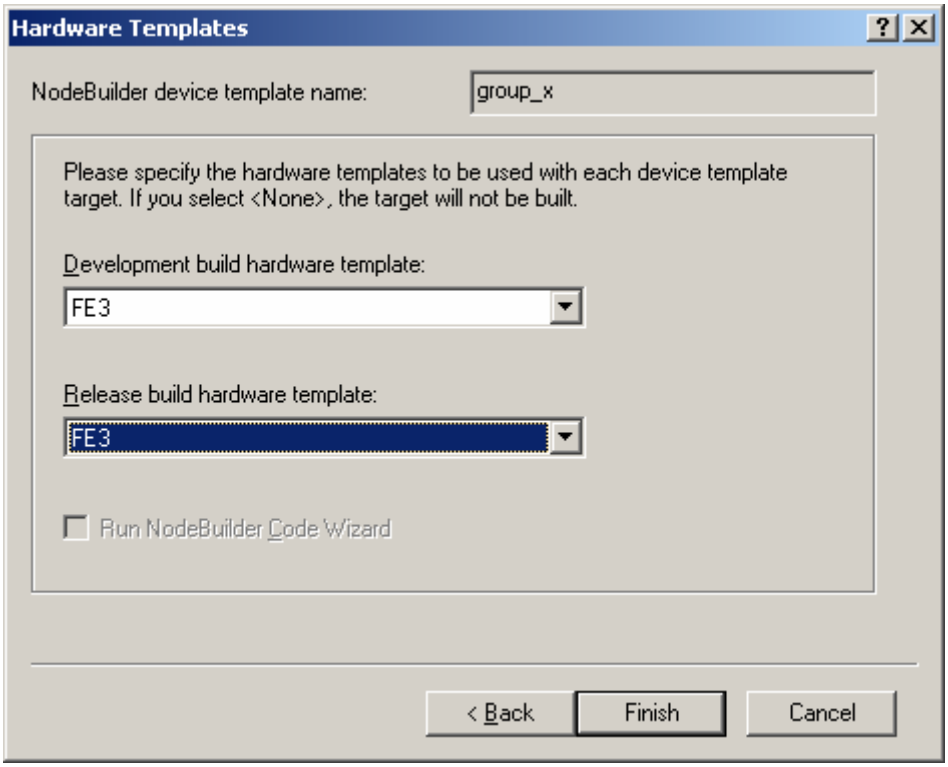


Kuva 9. Kirjoita laitemallin nimeksi ”group_x”, jossa x on ryhmänne numero. Aseta käännetyt tiedostot valmistumaan kotihakemistosi alihakemistoon ”\NeuronC\Output”.



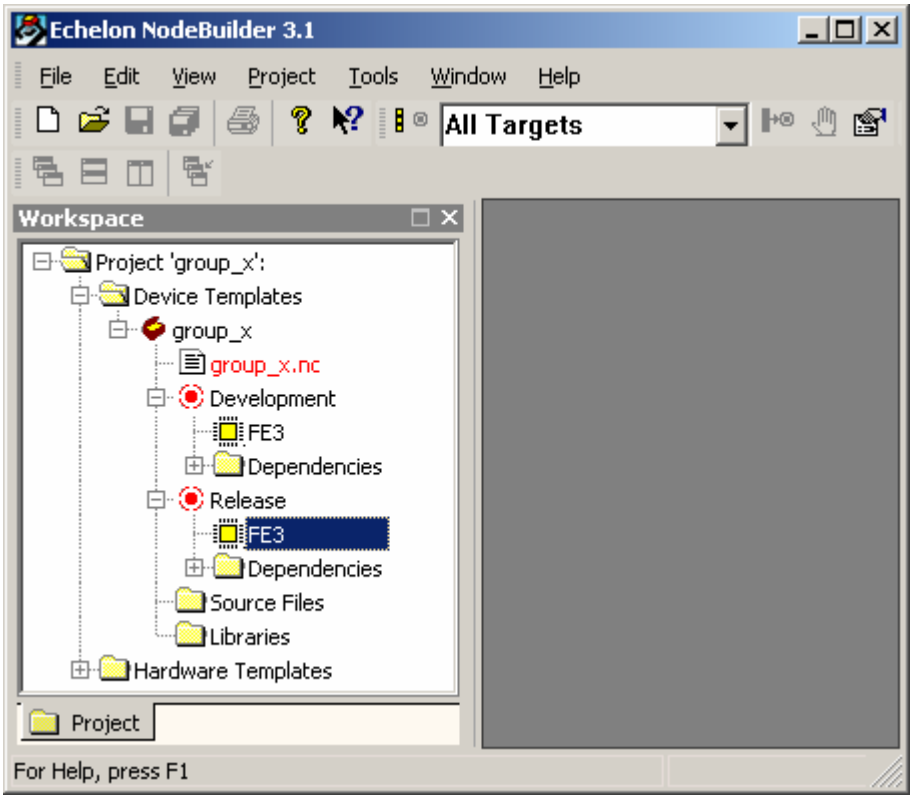
Kuva 10. Poista automaattinen tunnuksen luonti ja valitse tunnukseksi ”Non-standard”. Kirjoita sovellusohjelman tunnukseksi **MUL80121**

Tunnuskoodi MUL80121 sisältää Lon-verkon solmun tunnistetietoja Lon Node Image Loader -ohjelmalle. Siitä selviävät solmun nimi sekä laite- ja ohjelmistoversiot (COB/CEB/MUL - 8, hw v1, sw v21). Oikeaa tunnuskoodia käyttämällä vältytään ylimääräisiltä varoituksilta.



Kuva 11. Laitteiston mallitiedoston valinta.

Valitse laitteiston mallitiedostoksi FE3 kuvan 11. mukaisesti. Mallitiedosto sisältää tiedot simulointiympäristössä käytetystä Neuron-piirin versiosta, joka on Toshiba 3120FE5.

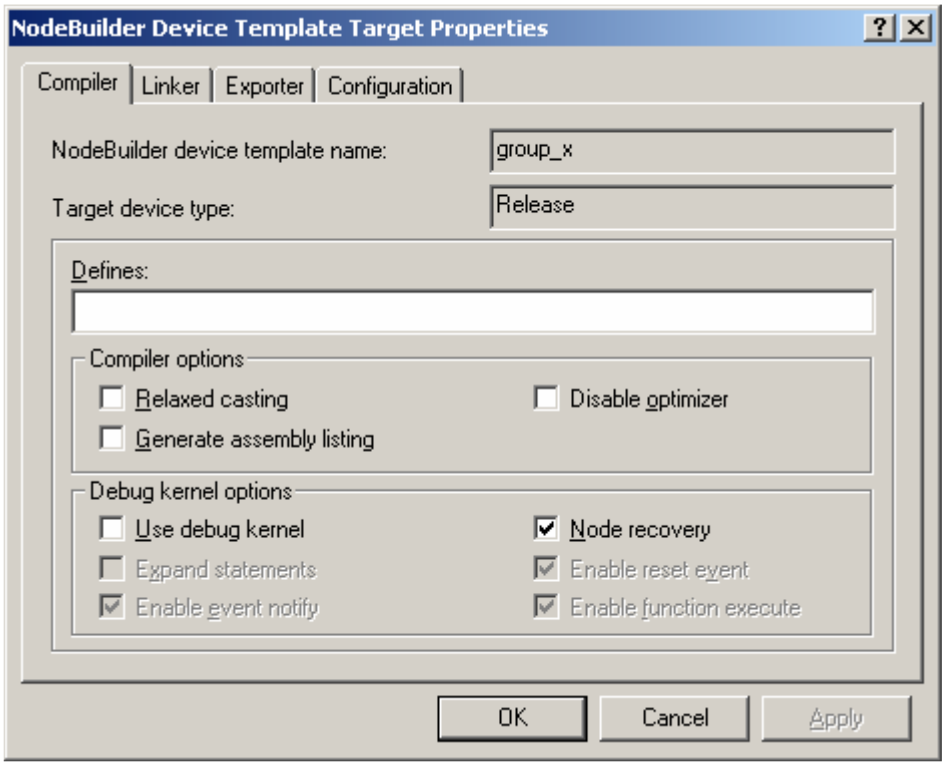


Kuva 12. NodeBuilderin automaattisesti luoma laitemalli.

Seuraavaksi NodeBuilder luo automaattisesti laitemallin (Device Template) ohjelmakoodiasi varten. Se sisältää valmiin rakenteen lähdekoodia, käytettyjä kirjastofunktioita ja laitteistotietoja varten. Voit tarkastella rakenteen sisältöä NodeBuilderin Workspace-ikkunassa, kuten kuvassa 12.

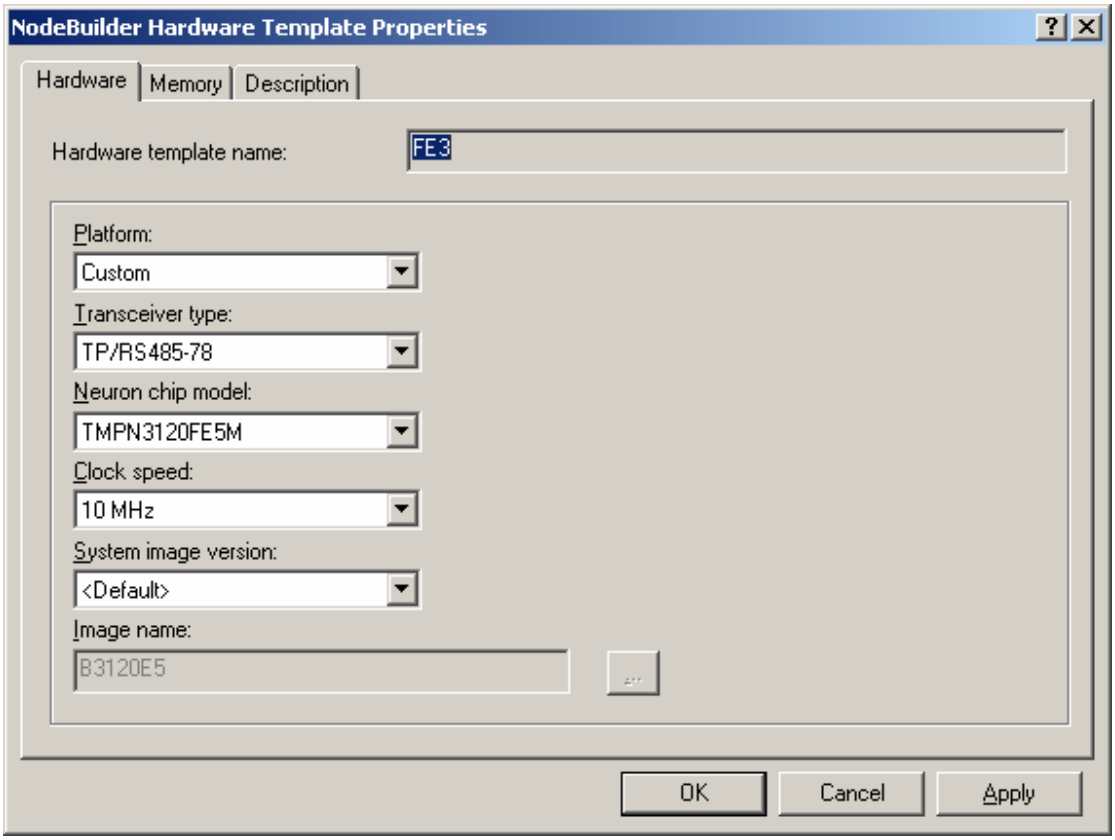
Laitemalli sisältää kääntäjän asetukset erikseen kehitys- ja julkaisuversiolle. Kehitysversio tukee debug-toimintoja, julkaisuversiosta sen sijaan karsitaan lopputuotteen kannalta tarpeettomat ominaisuudet ja sen kääntämisessä käytetään optimointialgoritmeja. NodeBuiderin yläreunassa olevasta vetopalkista voidaan valita, käännetäänkö molemmat versiot (All Targets) vai ainoastaan toinen niistä. Virheilmoitusten lukeminen vaikeutuu käännettäessä molemmat versiot kerralla, koska myös virheilmoitukset tulevat silloin kaksinkertaisina. Kehitysversion debug-toimintoja voidaan hyödyntää vain LonMaker-verkonhallintaohjelmalla, johon tutustutaan vasta kurssin viimeisessä harjoitustyössä. Kurssilla voidaan siis aina käyttää binääritiedoston julkaisuversiota, kunhan ”Node recovery” on kytketty päälle seuraavasti:

Napsauta NodeBuilderin Workspace-ikkunassa laitemallin julkaisuversiota ”Release”. Aseta ”Node recovery” päälle kuvan 13. mukaisesti. Tällöin kääntäjä lisää resetoinnin yhteyteen viiveen, joka auttaa piiriä selviytymään toistuvista vahtikoira-ajastimen raukeamisen aiheuttamista nollautumisista.



Kuva 13. Kytke laitemallin julkaisuversioon ”Node recovery” päälle.

Kuvassa 12. näkyy sinisenä maalattuna julkaisuversion laitteiston mallitiedosto. Tarkista hiiren oikealla näppäimellä, että sen asetukset vastaavat kuvaa 14. Asetukset ovat automaattisesti samat kohdissa Development, Release sekä Hardware User Templates.

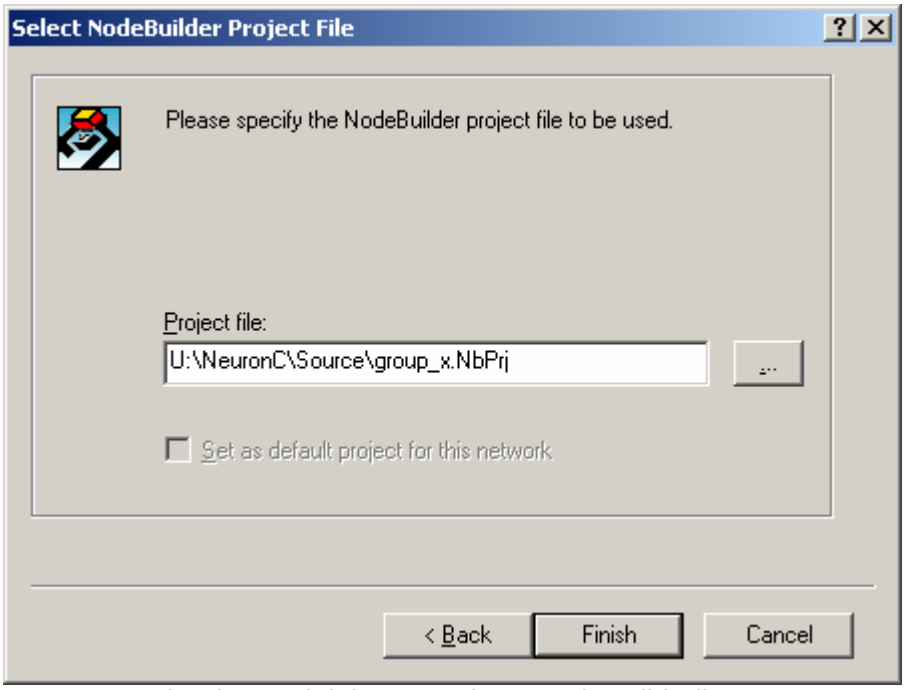


Kuva 14. Laitteiston mallitiedoston (Hardware Template) asetukset.

Kuvassa 12. Neuron C -lähdekooditiedosto, ”group_x.nc”, näkyy punaisena koska sitä ei ole vielä luotu. Tiedoston luomiseksi kopioi tiedosto “pohja_1.nc” ohjelmointi-PC:n hakemistosta ” D:\Mallit” kotihakemistossasi sijaitsevaan lähdekoodien hakemistoon ”\NeuronC\Source\group_x”, ja muuta sen nimi vastaamaan ryhmäänne, ”group_x.nc”. (x = ryhmänne numero). Tämä tiedosto sisältää pohjan, johon ensimmäinen harjoitustyö koodataan. Tiedoston saa näkyviin NodeBuilderin editointi-ikkunaan kaksoisnapsauttamalla tiedostonimeä Workspace-ikkunassa. Tämän jälkeen ohjelmakoodin editointi voidaan aloittaa.

Valmis ohjelmakoodi käännetään painamalla NodeBuiderin yläreunassa sijaitsevaa ”Build All”-nappia. Napin nimi viittaa kaikkiin projektin lähdekooditiedostoihin, sillä sen painaminen generoi binääritiedostoja vain vetopalkista valittuihin kohteisiin: Development, Release tai All Targets. Binääritiedostot syntyvät asetetun hakemistopolun mukaisesti omiin alihakemistoihinsa ”\NeuronC\Output\Development” ja/tai ”\NeuronC\Output\Release”.

Ohjelmakoodia käännettäessä avautuu ”Results Pane”-ikkuna, johon tulostuu listaus kääntäjän ilmoituksia. Sama listaus löytyy kohdehakemistosta .log -päätteisenä tekstitiedostona. Mikäli ohjelmakoodin kääntäminen keskeytyy virheen takia, ilmoittaa kääntäjä virheen syyn ja sijainnin. Kaksoisnapsauttamalla syyn tai sijainnin ilmaisevaa lausetta siirtää NodeBuilderin editori kohdistimen kyseiselle riville ohjelmakoodissa. Sovellusohjelman kirjoituksen tavoitteena on saada kääntäjä ilmoittamaan: ”0 error(s), 0 warning(s)”. Ilmoitus ”1 excluded” tarkoittaa, että kehitys- tai julkaisuversio jätettiin kääntämättä tehdyn asetuksen mukaisesti.



Kuva 15. Jo luodun projektin avaaminen NodeBuilderiin.

Seuraavalla kerralla käynnistettäessä NodeBuilder jatketaan jo luodun projektin kehittämistä. Tällöin haetaan projektitiedosto ryhmän kotihakemistosta, kuten kuvassa 15.

L6.2.2 Ohjelmointi-PC:n kytkeminen simulointiympäristöön

Ennen kuin ohjelmointi-PC:ssä toimivalla verkonhallintaohjelmalla tavoitetaan Lon-verkon solmut, on verkkosovitin kytkettävä haluttuun Lon-verkon kohtaan, simulointilaitteistoon on kytkettävä käyttöjännite sekä SLTALink Manager -ohjelma on käynnistettävä.

Fyysinen liityntä simulointiympäristön Lon-verkkoon tehdään SLTA-verkkosovittimen liitosjohdolla, joka voidaan kytkeä kolmeen eri paikkaan simulointilaitteistoon. Liityntäpiste valitaan sen mukaan, mitä laitteiston sisältämiä Neuron-piirejä halutaan tavoittaa. Verkonhallintaohjelmalla ei nähdä yhdyskäytävän toisella puolella olevia Neuron-piirejä. Yhdyskäytävän sisältämistä Neuron-piireistä nähdään se, kumman puoleiseen Lon-verkon osaan verkkosovittimella on liitytty.

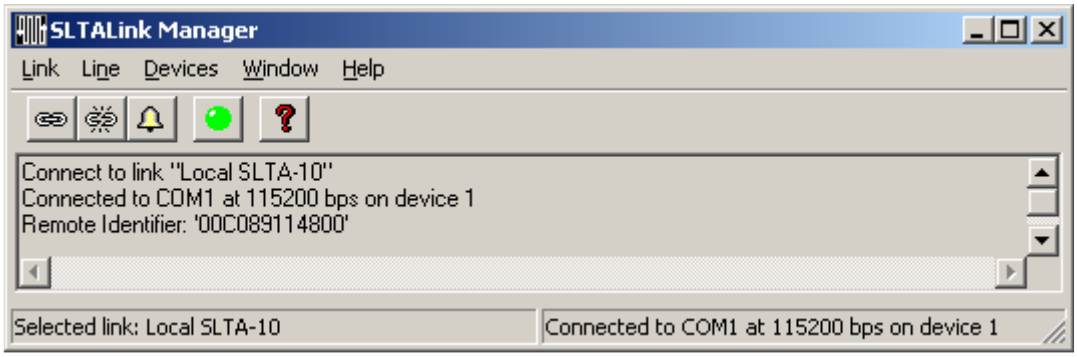
Simulointiympäristön päälaitteiston vasemmanpuoleisen pöytäsimulaattorin päällä on tunnuksella ”XG5” merkitty liitin. Tästä liittimestä tavoitetaan kuvaan 2. keltaisella merkityt Neuron-piirit. Oikeanpuoleisen pöytäsimulaattorin liittimestä tavoitetaan vastaavasti vaaleanpunaisella merkityt Neuron-piirit. Päälaitteiston takana sijaitseva standardimuuttujien kokeiluympäristö sisältää oman liityntäpisteen, joka sijaitsee laitekotelon vasemmassa kyljessä. Siitä tavoitetaan tämän laitteiston sisältämät kaksi Neuron-piiriä, jotka on merkitty kuvaan 2. vaalean vihreällä värillä.

Ennen SLTALink Manager -ohjelman käynnistystä tulee kytkeä käyttöjännite simulointilaitteistoon, koska verkkosovitin saa käyttöjännitteensä simulointilaitteistolta samaa johtonippua pitkin, jolla se kytkeytyy simulointilaitteiston Lon-verkkoon. Simulointilaitteiston sivussa on pääkytkin, ja pöytäsimulaattoreiden sekä standardimuuttujien kokeiluympäristön päällä on merkinnällä ”POWER” varustettu kytkin. Lisäksi ryhmä-PC:ssä on oma kytkin, ja se kytketään ainoastaan tarvittaessa viimeisenä laitteena ja sammutetaan ensimmäisenä. Katso kuvaa 1.

L6.2.3 SLTALink Manager

SLTALink Manager -ohjelma toimii ohjelmointi-PC:ssä verkkosovittimen ajurina, joka muodostaa yhteyden verkonhallintaohjelman ja SLTA-verkkosovittimen välille. Tämän ohjelman tulee olla aina päällä ja yhteyden kytkettynä, kun ohjelmointi-PC:n halutaan näkevän Lon-verkkoon. Vastaavasti ohjelman muodostama yhteys tulee katkaista ennen kuin Lon-laitteiston käyttöjännitteet katkaistaan. Verkkosovitin saa käyttöjännitteensä samaa johtoa pitkin, jolla se liittyy Lon-verkkoon.

Ohjelman kuvake on Windowsin tehtäväpalkin ilmoitusalueessa. Connect-nappia painamalla vaihtuu link statistics -valo punaisesta vihreäksi. Mahdolliset virheilmoitukset tekstikentässä eivät välttämättä estä toimintaa, kunhan valo on vihreä.



Kuva 16. SLTALink Manager -ohjelma on yhteydessä Lon-verkkosovittimeen.

L6.2.4 Lon Node Image Loader

Lon Node Image Loader -verkonhallintaohjelmalla siirretään käännetty Neuron C -ohjelmakoodi Neuron-piiriin. Sillä voidaan myös antaa verkonhallinnan käskyjä Lon-verkon solmuille, tarkkailla näiden toimintatilaa sekä lukea verkkomuuttujien arvoja. Verkkomuuttujien arvojen lukemiseen palataan harjoitustyössä 2.

Verkonhallintaohjelman vaatiman yhteyden luomiseksi ohjelmointi-PC kytketään fyysisesti verkkosovittimen liitosjohdolla simulointiympäristön Lon-verkkoon, jonka jälkeen simulointilaitteistoon kytketään käyttöjännite. Ohjelmointi-PC:ssä käynnistetään SLTALink Manager -ajuriohjelma, ja tämän jälkeen Lon Node Image Loader. Se kytkeytyy verkkoon Connect-nappia painamalla, jolloin Find nodes -napit aktivoituvat.

Lon-verkon solmuja voidaan etsiä automaattisesti tai Neuronin Service Pin -viestillä. Automaattisessa etsinnässä ”Nodes found”-lukema juoksee ylöspäin, kunnes haku keskeytetään Quit-napilla haluttujen solmujen löydyttyä. Kuva 17. esittää näkymän Lon Node Image Loader -ohjelmaan, kun verkkoon on kytkeydytty vasemmanpuoleisen pöytäsimulaattorin liittimestä ”XG5” ja on käytetty automaattista hakua.

Verkon solmut tunnistetaan sovellusohjelman tunnuksen (ID string) perusteella. Tämä tunnus on osa Neuron-piirin sisältämää sovellusohjelmaa ja ohjelmakoodin kirjoittaja asettaa sen ohjelmointityökaluun. Piirin sisältämää tunnusta voidaan siis muuttaa, mutta työkurssilla tämä tehdään vain viimeisessä harjoitustyössä siirryttäessä standardien verkkomuuttujien ja LonMaker-verkonhallintaohjelman käyttöön. Simulointiympäristön päälaitteistossa käytetään aina kuvan 17. mukaisia tunnuksia.

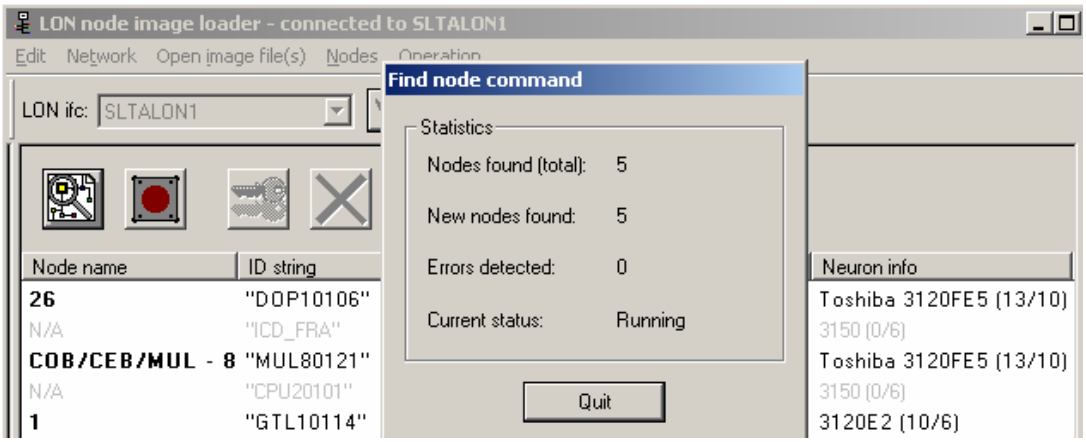
MUL80121 on koripaneelin Neuronin sovellusohjelman tunnus. Koripaneelissa käytetään aina tätä tunnusta. Tunnusta käytetään myös simulointiympäristön erillisessä osassa työkurssin ensimmäisissä harjoituksissa. Kuvan 17. oikeasta reunasta nähdään myös Neuron-piirin tyyppi. Työkurssilla sovellusohjelmia ladataan vain Multiboard-piirilevyille, joiden Neuron-piirit ovat tyyppiä Toshiba 3120FE5.

DOP10106 on kohdekutsupaneelin tunnus. Tähän piiriin ladataan sovellusohjelma kohdekutsuja käsittelevässä harjoitustyössä, jolloin myös uuden sovelluksen tulee sisältää tämä tunnus. Neuron-piirin tyyppi on jälleen Toshiba 3120FE5. Kohdekutsupaneeli on kytketty vasemman hissin kuiluverkkoon, joten sen Neuron-piiri tavoitetaan ainoastaan kytkeydyttäessä Lon-verkkoon vasemmanpuoleisen pöytäsimulaattorin liittimestä.

CPU20101 on hissi-CPU:n tunnus. Hissi-CPU sisältää erillisen sovellusprosessorin, ja Neuron-piiri toimii verkkoprosessorina. Tällöin piirin tyyppinä on 3150. Korikutsuharjoituksessa koripaneelista lähetettyjen viestien kohdeosoitteeksi asetetaan hissi-CPU:n verkko-osoite, joka selvitetään verkonhallintaohjelmalla.

GTL10114 on yhdyskäytävän tunnus. Kohdekutsuharjoituksessa kutsupaneelista lähetettyjen viestien kohdeosoitteeksi asetetaan yhdyskäytävän verkko-osoite, joka selvitetään verkonhallintaohjelmalla. Yhdyskäytävä on verkonhallintaohjelman ikkunassa aktiivisessa tilassa, vaikka siihen ei ole tarkoitus koskaan ladata mitään. ÄLÄ LATAA SOVELLUSOHJELMAA YHDYSKÄYTÄVÄN NEURON-PIIRIIN.

ICD_FRA on I-Link -näytön tunnus. Näyttö sisältää erillisen sovellusprosessorin, ja Neuron-piiri toimii verkkoprosessorina. Tällöin piirin tyyppinä on 3150. I-Link -näyttö on kytketty vasemman hissin koriverkkoon, joten sen Neuron-piiri tavoitetaan ainoastaan kytkeydyttäessä Lon-verkkoon vasemmanpuoleisen pöytäsimulaattorin liittimestä.



Kuva 17. Automaattinen Lon-verkon solmujen etsintä.

Lon-verkon solmu löytyy automaattisella haulla, mikäli etsittävän solmun Neuron-piiri sisältää Lon Node Image Loader -ohjelman olettamat verkkoasetukset. Kurssin harjoitustöissä lähdetään liikkeelle mahdollisimman yksinkertaisesta Neuron C -sovelluksesta, jossa ei vielä tehdä verkkoasetuksia. Tällöin solmu voidaan löytää ainoastaan painamalla ensin Lon Node Image Loaderin nappia ”Find nodes via service pin”, ja sitten fyysistä Service Pin -nappia operoitavan Neuronin piirilevyllä aivan Neuron-piirin vieressä. Sisältämästään sovelluksesta tai verkkoasetuksista riippumatta Neuron-piiri reagoi Service Pin -painallukseen lähettämällä yleislähetyksenä viestin, joka sisältää piiriin valmistuksen yhteydessä pysyvästi kirjoitetun NeuronID-tunnuksen sekä sovellusohjelman tunnuksen.

Jotta binääriksi käännetty Neuron C -sovellus saadaan siirrettyä Neuron-piirille, tulee NodeBuilderin tuottama image-tiedosto hakea Lon Node Image Loaderin vasemmalla sijaitsevaan images-ikkunaan. NodeBuilderilla tuotetut binääritiedostot löytyvät projektikohtaisten asetusten mukaisesti esim. hakemistoista ”U:\NeuronC\Output\Release\” tai/ja ” U:\NeuronC\Output\Development\”, riippuen kääntäjän asetuksista. Haluttaessa voidaan tuottaa yhdellä kertaa ohjelman kehitys- ja julkaisuversiot, joiden kääntämisessä on käytetty eri parametreja. Kehitysversioon voidaan esimerkiksi sisällyttää debug-ominaisuuksia. Haettavan binääritiedoston nimen alkuosa on sama kuin projektin nimi NodeBuilderissa, ja loppuosana on NEI.

HUOMIO: Aina, kun NodeBuilderilla käännetään Neuron C -koodi Neuron-piirille siirtoa varten, tulee käännetty binääritiedosto hakea uudelleen Lon Node Image Loaderiin. Lon Node Image Loader siirtyy uuteen binääritiedostoon, kun images-ikkunassa näkyvää aiemmin haetun binääritiedoston nimeä kaksoisnapsautetaan. Päivittyminen huomataan tiedostonimen lihavoinnin katoamisesta. Ohjelma tunnistaa, jos tarjolla olisi uudempikin binääritiedosto, ja ilmaisee tämän lihavoimalla tiedoston nimen. Se ei siis kuitenkaan automaattisesti siirry tähän uudempaan versioon.

Binääri-image voidaan siirtää Neuron-piiriin Download-napilla, kun sekä binääritiedosto että ohjelmoitava Neuron ovat valittuina eli niiden nimet näkyvät sinisellä pohjalla. Siirron päätyttyä tulee tarkistaa, että siirto onnistuu virheettä, kuten kuvassa 18. Tarvittaessa siirto tulee toistaa, jotta päästään virheettömään lopputulokseen.

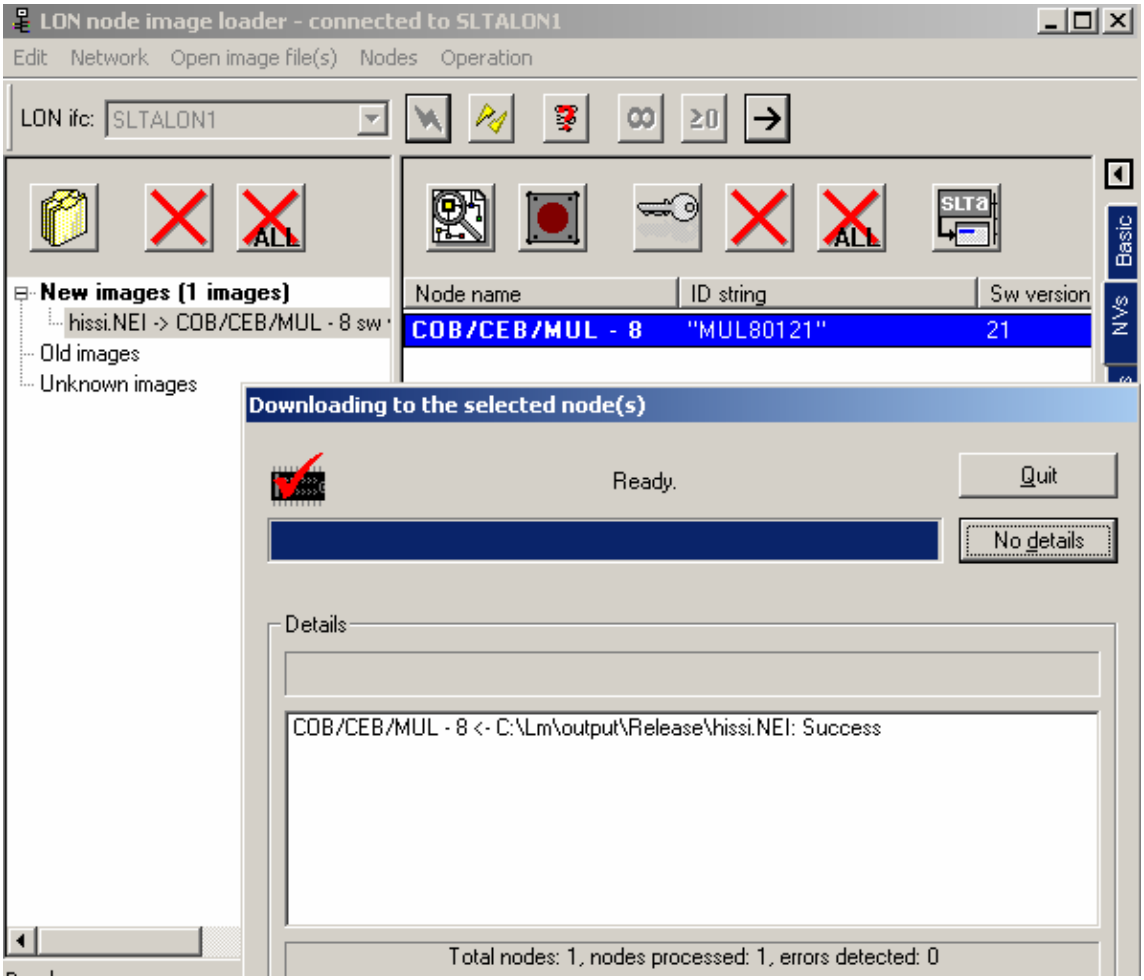
Siirron päätyttyä Neuron on soft offline -tilassa. Takaisin online-tilaan sen saa napsauttamalla hiiren oikealla napilla kyseisen Neuronin tunnuspalkkia Lon Node Image Loaderissa, ja valitsemalla avautuvasta valikosta ”Online”. Piirin tila voidaan tarkastaa ”Status”-välilehdestä, joka löytyy ohjelman oikeanpuoleisesta alareunasta. Mikäli siirron kohteena ollut Neuron on jumiutunut tai hissi-CPU on ajautunut häiriötilaan, on suoritettava koko laitteiston resetointi käyttöjännite pois kytkemällä.

Resetointi

Koko laitteiston resetointi käyttöjännite pois kytkemällä tapahtuu seuraavasti:

1. Paina ”Disconnect” Lon Node Image Loaderissa.
2. Paina “Disconnect” SLTALink Managerissa.
3. Katkaise käyttöjännite, odota hetki ja kytke käyttöjännite takaisin.
4. Paina “Connect” SLTALink Managerissa.
5. Paina ”Connect” Lon Node Image Loaderissa.

Mikäli Lon-verkon kokoonpano säilyy resetoinnissa ennallaan, ei solmuja tarvitse hakea uudelleen Lon Node Image Loaderiin, vaan toimintaa voidaan jatkaa vanhoilla verkkoasetuksilla.



Kuva 18. Binääri-imagen onnistunut siirto Neuron-piirille.

L6.2.5 LonMaker Integration Tool

Echelonin NodeBuilder-ohjelmistopaketin mukana tulee oma verkonhallintatyökalu, LonMaker. Tämä ohjelma on laajassa käytössä, ja mm. asentajat käyttävät sitä muodostaessaan Lon-verkkoja standardin liittymäpinnan sisältävistä komponenteista rakennusautomaatiossa ja teollisuudessa.

Verkon yhdistämistä LonMaker Integration Tool -ohjelmaa käyttäen tarkastellaan kurssin viimeisessä harjoitustyössä. Siinä otetaan käyttöön simulointiympäristön erillinen laitteisto, jossa voidaan kokeilla standardien verkkomuuttujien generointia ja luotujen Lon-verkon solmujen yhdistämistä graafisessa ympäristössä.

L6.3 Sovelluskehityksen turvaohjeet

Valmiiden Lon-verkkossa toimivien laitteiden, kuten antureiden ja toimilaitteiden, verkkoasetukset kirjoitetaan Neuron-piireihin tyypillisesti asennuskohteessa asentajien toimesta. Tätä työvaihetta suorittaa laaja joukko ihmisiä, ja työvaiheen tekeminen on suunniteltu mahdollisimman sujuvaksi ja varmaksi.

Neuron-piiriä kehitettäessä ei sen sijaan ole tarkoitettu, että loppukäyttäjä kirjoittaisi piirin varsinaista sovellusohjelmaa. Sovellusohjelman kehityksen on oletettu tapahtuvan harjaantuneiden ammattilaisten toimesta muutamissa Lon-verkon laitteita kehittävissä yrityksissä. Kehitystyökaluja luotaessa niiden käytön turvallisuus ei ole ollut etusijalla, vaan vastuuta on jätetty kehitystyökalujen käyttäjille.

Sovellusohjelma sisältää Neuron-piirin toiminnalle kriittisiä laitteiston ja verkkoliikenteen asetusparametreja. Sovellusohjelmaa siirrettäessä nämä kriittiset parametrit tallennetaan Neuron-piirin EEPROM-muistiin. Irrallaan olevaan Neuron-piiriin voidaan sovellusohjelma ja asetusparametrit kirjoittaa erityisellä ohjelmointilaitteella. Mutta kun Neuron-piiri on kerran juotettu laitteen piirilevyyn, onnistuu sovellusohjelman ja asetusparametrien muuttaminen ainoastaan Lon-verkkoa pitkin. Tämän yhteyden toimiminen puolestaan edellyttää, että asetukset ovat kohdallaan. Neuron-piiri voidaan siis helposti ”hukata” niin, ettei siihen enää saada yhteyttä verkon välityksellä. Jotta näin ei kävisi, noudatamme seuraavia toimenpiteitä:

- Jaa suorittamasi tehtävä osiin, jotka ratkaiset yhden kerrallaan. Jokaisen vaiheen yhteydessä varmistu koodisi toiminnasta.

- Työskentele systemaattisesti ja tallenna lähdekoodit jokaisesta sovellusohjelmasta, jonka siirrät Neuron-piirille. Dokumentoi ne lyhyesti, esim. minkä muutoksen teit edelliseen versioon verrattuna.

Varoituksen käyttämättä jäävästä muuttujasta tai funktiosta saa pois kääntäjän käskyllä #pragma ignore_notused muuttujan_nimi tai #pragma ignore_notused funktion_nimi. Käsky sijoitetaan heti käyttämättä jäävän määrittelyn jälkeen.

- Käännetty binääritiedosto siirretään Neuron-piirille vain, mikäli kääntäjä ei anna virheilmoituksia eikä varoituksia. Mikäli et saa varoituksia pois, pyydä assistentilta apua!

Neuron on arka ikuisille silmukoille, koska se on luonteeltaan ei-keskeyttävä (non pre-emptive). Tällaisessa järjestelmässä tehtävän vaihto onnistuu ainoastaan suorittavan tehtävän niin salliessa.

- Tarkista aina ennen binäärikoodin latausta sovelluksen sisältämien silmukoiden indeksit. Varmistu, että silmukan päättymisehto toteutuu, eli että ohjelman kulku pääsee jatkumaan silmukasta eteenpäin. Kääntäjä ei tätä osaa tarkistaa, joten ohjelmakoodi voi rikkoa piirin vaikka virhe- ja varoitusilmoituksia ei käännettäessä tulisikaan.

Neuron-piirin EEPROM-muisti sisältää Neuron-piirin toiminnalle kriittisiä laitteiston ja verkkoliikenteen asetusparametreja. Kaikki Neuron-piirin muistityypit käyttävät samaa 16-bittistä osoiteavaruutta ja niihin viitataan samoilla mekanismeilla. Niinpä kirjoitusoperaatio voi kohdistua RAM-muistin sijasta EEPROM-muistiin, mikäli osoitteen lukuarvo on virheellinen. Väärälle muistialueelle kirjoittamisen välttämiseksi:

- Nimeä jokainen määrittelemäsi objekti yksilöllisesti, myös esim. eri struct-tyyppien alikentissä. Vaikka sovelluksen kirjoittajan mielestä sekaannusta ei pitäisi tapahtua, niin kääntäjä saattaa tehdä väärän tulkinnan.

- Vältä osoitinrakenteita ja pyri käyttämään muuttujia suoraan, ilman että niihin tarvitsee viitata osoittimien avulla. Hyödynnä structeja, ne soveltuvat monimutkaisemmankin datan käsittelyyn.

- Jos luot osoitinmuuttujan, alusta se osoittamaan johonkin käyttämääsi objektiin. Ennen alustusta se voi osoittaa minne tahansa.

Verkkoliikenteen asetustaulukoihin kirjoitetaan varusohjelmiston funktioilla ”update_domain( )”, ”update_address( )” sekä ”update_nv( )”. Näiden funktioiden argumenttina käytetään struct-muuttujaa, jonka kenttien arvoja muutetaan ennen EEPROM-muistin asetustaulukkoon kirjoitusta. Alustamatta jäävä kenttä voi aiheuttaa virheellisten tietojen kirjoittamisen asetustaulukkoon.

- Verkkoliikenteen asetustaulukkoon kirjoittaessasi alusta struct-muuttuja vastaavalla access-funktiolla ennen update-funktion käyttöä. Esim. käyttäessäsi ”update_domain( )”-funktiota alusta muuttuja ensin ”access_domain( )”-funktiolla. Näin kaikki muuttujan alikentät tulee varmasti alustettua.

Koska Neuron-piirin EEPROM-muisti sisältävää tärkeitä asetustietoja, on Neuron C:ssä oletuksena estetty näille muistialueille kirjoittaminen osoittimen välityksellä. Kääntäjä tulkitsee EEPROM-muistiin osoittavan osoittimen vakion osoittimeksi. EEPROM-muuttujan lisäksi myös verkkomuuttujan ja asetusparametrin osoitinta kohdellaan vakion osoittimena, mikä estää muutosten tekemisen osoittimen välityksellä. Tähän voidaan tehdä poikkeus kääntäjän käskyllä, mikä kuitenkin ohittaa kääntäjän varmistusmekanismit.

- Kurssilla ei käytetä kääntäjän varmistusmekanismeja ohittavia kääntäjän käskyjä.

Sisääntulon verkkopuskurin on oltava riittävän suuri, jotta piiri on hallittavissa verkonhallintaviesteillä myös uuden sovellusohjelman lataamisen jälkeen. Koko on oletuksena 66 tavua.

- Älä muuta sisääntulon verkkopuskurin kokoa.